BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF MATHEMATICS
ÚSTAV MATEMATIKY

SHAPE OPTIMIZATION OF THE MACHINE COMPONENTS DUE TO VARIABILITY OF INPUT DATA
OPTIMALIZACE TVARU STROJNÍCH SOUČÁSTÍ S VLIVEM VARIABILILTY VSTUPNÍCH ÚDAJŮ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Paranee Sawadkosin**
AUTOR PRÁCE

SUPERVISOR — **doc. Ing. Pavel Novotný, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2019

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Specification Master's Thesis

Department:	Institute of Mathematics
Student:	**Paranee Sawadkosin**
Study programme:	Applied Sciences in Engineering
Study field:	Mathematical Engineering
Leader:	**doc. Ing. Pavel Novotný, Ph.D.**
Academic year:	2018/19

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Master's Thesis:

Shape Optimization of the Machine Components due to Variability of Input Data

Concise characteristic of the task:

The thesis deals with the development of algorithms for optimization of turbocharger thrust bearing properties with influence of variability of input data. The work involves the development of optimization algorithms affecting variable structural, technological or operational characteristics in industrial applications. The algorithm have to be able to find the optimal parameter values to the selected problem. The work includes a development of program incorporating partial models in the programming language (Matlab, C ++, Python or Fortran) and their application to 1D / 2D hydrodynamic or structural problems.

Goals Master's Thesis:

Searching for optimization methods for a given problem.
Program for parameter optimization of turbocharger bearing systems.
Application of optimization methods to design dimensions of the turbocharger bearing.

Recommended bibliography:

LUKE, Sean. Essentials of metaheuristics. 2. ed., 2013. ISBN 9781300549628.

TVRDÍK, Josef. Evoluční algoritmy [online]. Ostrava, 2010 [cit. 2018-09-04]. Dostupné z: http://www1.osu.cz/~tvrdik/wp-content/uploads/XEVALG_10.pdf. Učební texty. Ostravská univerzita.

STACHOWIAK, Gwidon W. a BATCHELOR, Andrew W. Engineering Tribology. 3. vyd. Boston: Elsevier Butterworth-Heinemann, 2005. ISBN 0-7506-7836-4.

NGUYEN-SCHÄFER, Hung. Rotordynamics of Automotive Turbochargers. Second Edition. Ludwigsburg, Germany: Springer, 2015. ISBN 978-3-319-17643-7.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2018/19

In Brno,

L. S.

prof. RNDr. Josef Šlapal, CSc. Director of the Institute	doc. Ing. Jaroslav Katolický, Ph.D. FME dean

Abstract

The objective of this Master's thesis is to find shape optimal design based on minimizing friction force of thrust bearing by using genetic algorithm(GA) which is one of an optimization toolbox in Matlab. Reducing the friction force of thrust bearing is one way of making shaft to decreasing friction losses. With four parameters of thrust bearing geometry number of segments(m), angle of running surface(α), segment inner radius(R_0), and segment outer radius(R_1) substitute in Reynolds' equation. In order to know friction force, it is necessary to generate a connecting variable, oil film thickness(h_0) from loading capacity(W) and revolution per minute(rpm). Friction power loss, as well as weight function conclude the final shape optimization of thrust bearing: $m = 7$, $\alpha = 0.1°$, $R_0 = 15$ mm, and $R_1 = 20$ mm.

Keywords

shape optimization, friction force, thrust bearing, genetic algorithm, Matlab, Reynolds equation, friction power loss, weight function

SAWADKOSIN, P. *Shape Optimization of the Machine Components due to Variability of Input Data*. Brno: Brno University of Technology, Faculty of Mechanical Engineering, 2019. 70 p. Diploma thesis supervisor Assoc. prof. MSc. Pavel Novotný, Ph.D.

I declare that I have written this diploma thesis *Shape Optimization of the Machine Components due to Variability of Input Data* on my own under the direction of my supervisor, Assoc. prof. MSc. Pavel Novotný, Ph.D., and using the sources listed in the bibliography.

May 24, 2019 **Paranee Sawadkosin**

I would like to dedicate this thesis, to my parents, Thanapich and Thunjira, my brother Subhapich, my uncle Natthapat, my special one Sudarat, my beloved friends Natnicha and Anirut, my Chinese friend Windy, my colleague Ing. Jakub Kůdela all of these people who always support me. Without one of them I cannot achieve my goal today.

Paranee Sawadkosin

Contents

Preface

The thesis begins with Chapter 1 which is the introduction of this work. It starts with the background of the writer's Bachelor's Degree project. Then state the problem, ideas to apply, and the scope of work.

The Chapter 2 is devoted to hydrodynamics bearing analysis. First is the history how the theory of hydrodynamic lubrication was published by Reynolds. After will show how to generate Reynolds' equation and also the equations of pressure, loading capacity, friction force, and friction coefficient that will be essentially used in the following chapter.

The Chapter 3 Newton's method brings an idea how to find the root of the equation. Moreover, there are advantages and disadvantages about using this method.

The Chapter 4 *fzero* command in Matlab which is used to find the root of the equation. Therefore, it is interesting to know the ideas how it was developed and how to construct this command.

The Chapter 5 Genetic Algorithm shows the ideas to apply for optimization, optimization toolbox in Matlab, description of GAToolbox, and pros and cons of this algorithm at the end of the chapter.

The Chapter 6 Real Data Application will describe all the procedure of finding the objective of this thesis. An ideal way to find oil film thickness, there are Newton's method and *fzero*. Brute force algorithm and Genetic algorithm will be explained. Including the results when apply all ideas which mentioned in above chapters

The Chapter 7 Conclusion and Future Work represents how to conclude the shape optimization of this thrust bearing. For future work shows that there are some recommendations from the writer wish the reader to obtain more effective result.

Chapter 1

Introduction

1.1 Background

My friends and I had designed a part of a copy machine to satisfy customer's specification. We used design of experiment (DOE) to guide us while finding the parameters that effected the part. Then we found the optimal values of the parameters to obtain the dimension within the tolerance. During experiment, the company had lost not only the money in materials and some staff but also the ability to produce others products. Since then we hoped that there would be some method to solve this kind of problem in a wiser way.

1.2 Statement of the Problems

Thrust bearing is a part in the turbocharger machine which is used to withstand the axial force generated by the turbine shaft see Figure (1.1). The axial force acting on the thrust bearing is mainly caused by the imbalance between the turbine wheel and the compressor wheel [1]. Thrust bearing will support the precision of shaft in both radial and axial while operating and also minimize the friction force by allowing the shaft to rotate smoothly without any interruption. Since the friction losses will reduce the performance and the efficiency of turbocharger, so minimizing friction force is one of the alternatives to compensate and increase both performance and efficiency of turbocharger[1, 2].

Most of researchers solved their interesting on thrust bearing by using Computational Fluid Dynamics (CFD)[2] or Finite Difference Method (FDM)[3]. Erik had found that the axial force acting on the turbocharger thrust bearing which was measured by strain gauges and axial displacement and calculated with force balance is the most significant value that will influence to both oil film thickness and the frictional loss in the bearing [5]. These would be an assumption that the value of oil film thickness and the frictional loss are related to the loading capacity. Erland showed how to apply many of stochastic optimization techniques in Matlab and one of them is Genetic algorithm(GA). In order to find optimal design of tidal power generator, his objective function is to minimize the cost of the electrical machine. He set his parameters which would effect the mass of the machine and many of constraints to receive satisfied answer[4].

At our present work, we will apply the knowledge of both engineering and mathematics and the advantages of Matlab program to find shape optimization by minimizing the

friction force or by minimizing coefficient of friction force (i.e.,ratio of the friction and normal force acting on the surface)[4] on thrust bearing. However, the values of load capacity acting on this part are already fixed so we will focus on friction force during the computation. We believe that this can save time consuming while finding shape optimization, knowing the friction force of the geometry would be. There is no need to wait until we finish designing my part then do the calculation, and then go back to design process again if it is not satisfied.

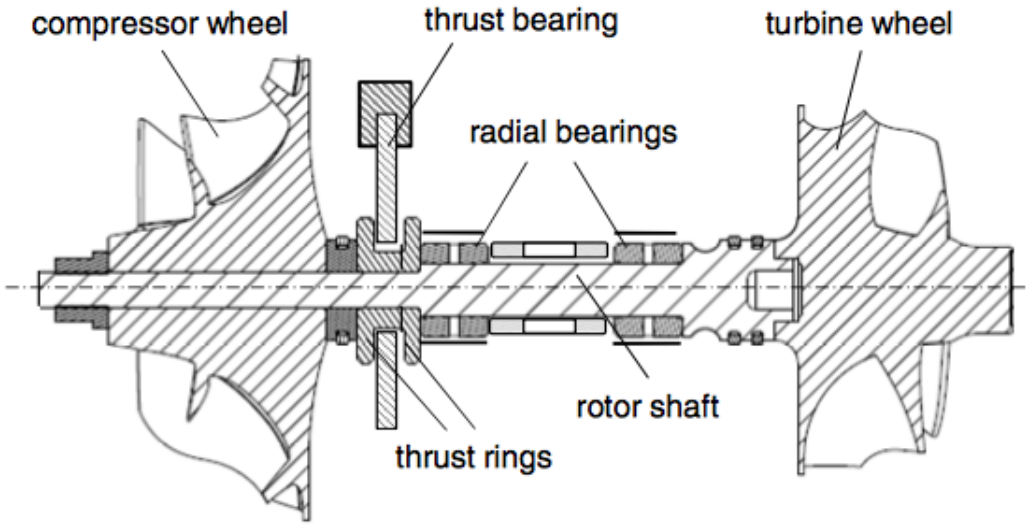


Figure 1.1: Thrust bearing in the turbocharger, featuring the compressor wheel, the turbine wheel, and a segment of bearing[6].

1.3 Scope of Work

1. There are four parameters with the following range see Figure (1.2)

   - Number of segments (m) = [3–12]
   - Angle of running surface (α) = [0.1–1] $^\circ$
   - Segment inner radius (R_0) = [13–15] mm
   - Segment outer radius (R_1) = [20–50] mm

2. Considering load on whole bearing with respect to revolutions per minute (rpm) in the Table 1.1. Even though there are thirteen states of loading capacities, for the final shape optimization would be the only one value for each four parameters that be optimal for all states.

3. Let constant viscosity(η) equals to 0.01 [Pa.s]

4. Using Matlab program to find optimal shape of the thrust bearing

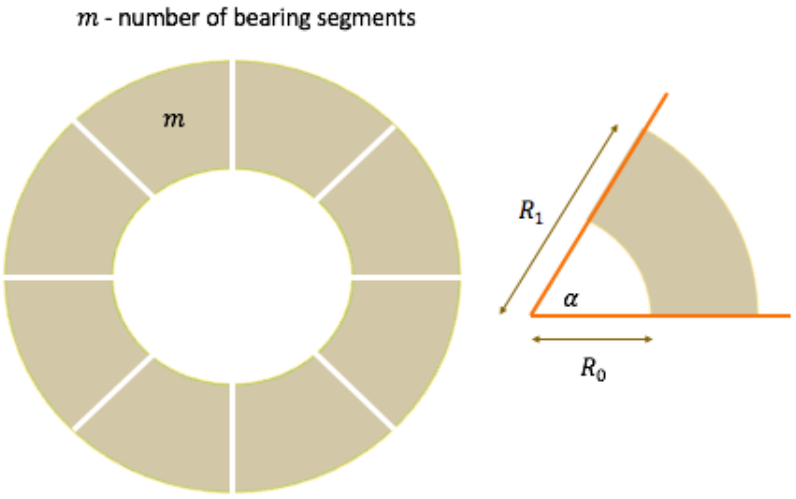


Figure 1.2: Represent four parameters for designing thrust bearing

State	Loading Capacities[N]	rpm[1/min]
1	220	20000
2	256	22500
3	343	25000
4	462	27500
5	567	30000
6	657	32500
7	772	35000
8	899	37500
9	1012	40000
10	1059	42500
11	994	45000
12	537	47500
13	100	50000

Table 1.1: Loading capacities[N] and rpm[1/min]

Chapter 2

Hydrodynamics Bearing Analysis

2.1 History of Hydrodynamic Lubrication

"*An engineer Beauchamp Tower had noticed an oil leaking out of the hole in journal bearing. By placing wooden bung in the hole, he realized that there was a pressure in the oil which could separate the sliding surface by a hydraulic force. Fortunately, during that time when he discovered many hydrodynamic theories of lubrication, Osborne Reynolds could use Tower's observed data to provide experimental support of his hydrodynamic lubrication. In 1866, Reynolds published a theory of hydrodynamic lubrication. He proved that a viscous liquid can physically separate two sliding surfaces by hydrodynamic pressure, resulting in low friction. And at the beginning of 20th century, Michell and Kingsbury had successfully applied the theory of hydrodynamic lubrication to thrust bearing and the pivoted pad bearing was developed as an outcome.*"[7]

2.2 Simplifications to the Reynolds Equation

'Reynolds equation' is used to express mathematically in the form of all hydrodynamic lubrication. It is mostly derived by considering the equilibrium of an element of liquid subjected to viscous shear and applying the continuity of flow principle.

2.2.1 Equilibrium of an Element

Consider a small element of fluid from a hydrodynamic film shown in Figure (2.1). For simplicity, assume that the forces on the element are acting initially in the 'x' direction only. Since the element is in equilibrium, forces acting to the left must balance the forces acting to the right, so Since the element is in equilibrium, forces acting to the left must balance the forces acting to the right, so

$$pdydz + (\tau_{\mathrm{x}} + \frac{\partial \tau_{\mathrm{x}}}{\partial z}dz)dxdy = (p + \frac{\partial p}{\partial x})dydz + \tau_{\mathrm{x}}dxdy \tag{2.1}$$

which after simplifying gives:

$$\frac{\partial \tau_{\mathrm{x}}}{\partial z}dxdydz = \frac{\partial p}{\partial x}dxdydz \tag{2.2}$$

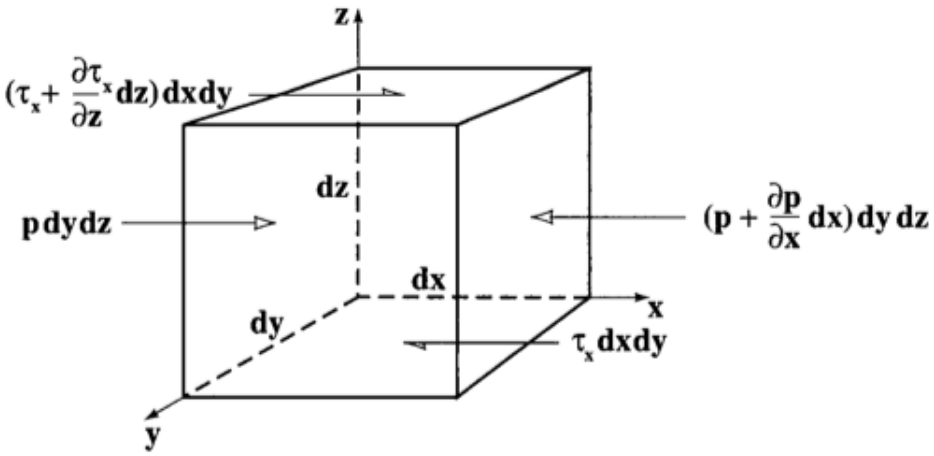


Figure 2.1: Equilibrium of an element of fluid from hydrodynamic film; p is the pressure, τ_x is the shear stress acting in the x direction[7].

Assuming that $dxdydz \neq 0$ (i.e. non zero volume), both sides of equation (4.2) can be divided by this value and then the equilibrium condition for forces acting in the 'x' direction is obtained,

$$\frac{\partial \tau_x}{\partial z} = \frac{\partial p}{\partial x} \tag{2.3}$$

A similar exercise can be performed for the forces acting in the 'y' (out of the page) direction, yielding the second equilibrium condition,

$$\frac{\partial \tau_y}{\partial z} = \frac{\partial p}{\partial y} \tag{2.4}$$

In the 'z' direction since the pressure is constant through the film, the pressure gradient is equal to zero:

$$\frac{\partial p}{\partial z} = 0 \tag{2.5}$$

It should be noted that the shear stress in expression (2.3) is acting in the 'x' direction while in expression (2.4) it is acting in the 'y' direction, thus the values of the shear stress in these expressions are different.
The shear stress τ can be expressed in terms of dynamic viscosity and shear rates:

$$\tau_x = \eta \frac{u}{h} = \eta \frac{\partial u}{\partial z} \tag{2.6}$$

where $\tau_x$is the shear stress acting in the 'x' direction [Pa].
Substituting (2.6) into (2.3), the equilibrium condition for the forces acting on x direction is obtained:

$$\frac{\partial p}{\partial x} = \frac{\partial}{\partial z}\left(\eta \frac{\partial u}{\partial z}\right) \tag{2.7}$$

Along the 'y' (out of the page) direction, however, the velocity is different and consequently the shear stress is different:

$$\tau_{\mathrm{y}} = \eta \frac{v}{h} = \eta \frac{\partial v}{\partial z} \tag{2.8}$$

where τ_{y} is the shear stress acting in the 'y' direction [Pa] and v is the sliding velocity in the 'y' direction [m/s]. Substituting (2.8) into (2.4), the equilibrium condition for the forces acting on x direction is obtained:

$$\frac{\partial p}{\partial y} = \frac{\partial}{\partial z}\left(\eta \frac{\partial v}{\partial z}\right) \tag{2.9}$$

Equations (2.7) and (2.9) can now be integrated following the assumption that the viscosity η is constant throughout the film. For example, considering equation (2.7) in 'x' axis:

$$\frac{\partial p}{\partial x} \partial z = \partial \left(\eta \frac{\partial u}{\partial z}\right)$$

and integrating gives:

$$\frac{\partial p}{\partial x} z + C_1 = \eta \frac{\partial u}{\partial z}$$

Separating variables again,

$$\left(\frac{\partial p}{\partial x} z + C_1\right) \partial z = \eta \partial u$$

and integrating again yields:

$$\frac{\partial p}{\partial x} \frac{z^2}{2} + C_1 z + C_2 = \eta u \tag{2.10}$$

And there is no slip or velocity discontinuity between liquid and solid at the boundaries of the wedge, the boundary conditions are:

$$\begin{aligned} u &= U_2 \qquad at \qquad z = 0 \\ u &= U_1 \qquad at \qquad z = h \end{aligned}$$

In the general case, there are two velocities corresponding to each of the surfaces 'U_1' and 'U_2'. By substituting these boundary conditions into (2.10) the constants 'C_1' and 'C_2' are calculated:

$$\begin{aligned} C_1 &= (U_1 - U_2)\frac{\eta}{h} - \frac{\partial p}{\partial x}\frac{h}{2} \\ C_2 &= \eta U_2 \end{aligned}$$

Substituting these into (2.10) yields:

$$\frac{\partial p}{\partial x}\frac{z^2}{2} + (U_1 - U_2)\frac{\eta z}{h} - \frac{\partial p}{\partial x}\frac{hz}{2} + \eta U_2 = \eta u$$

Dividing and simplifying gives the expression for velocity in the 'x' direction:

$$u = \left(\frac{z^2 - zh}{2\eta}\right)\frac{\partial p}{\partial x} + (U_1 - U_2)\frac{z}{h} + U_2 \tag{2.11}$$

In a similar manner a formula for velocity in the 'y' direction is obtained.

$$v = \left(\frac{z^2 - zh}{2\eta}\right)\frac{\partial p}{\partial y} + (V_1 - V_2)\frac{z}{h} + V_2 \tag{2.12}$$

The three separate terms in any of the velocity equations (2.11) and (2.12) represent the velocity profiles across the fluid film and they are schematically shown in Figure (2.2).

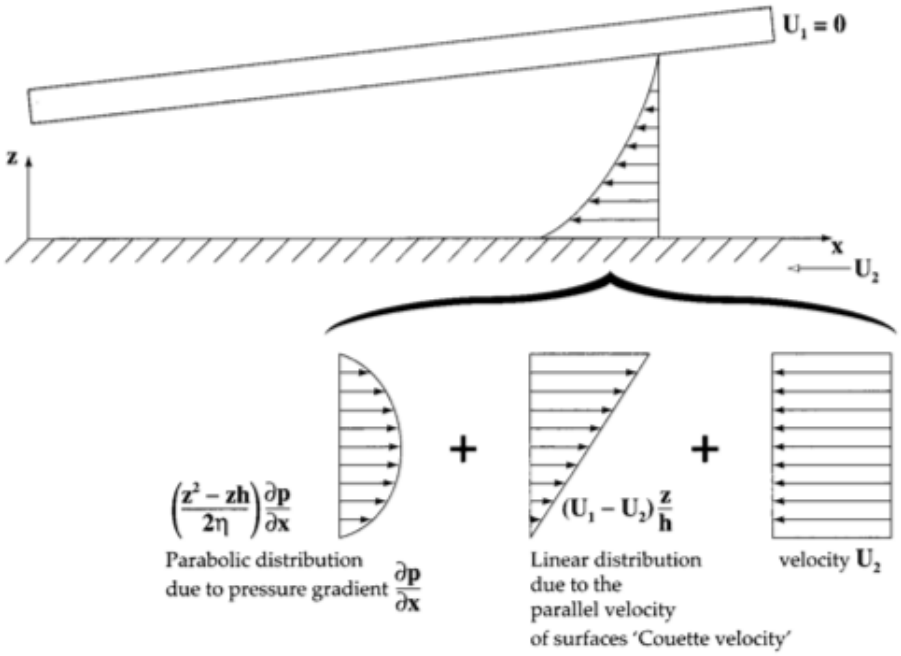


Figure 2.2: Velocity profiles at the entry of the hydrodynamic film[7].

2.2.2 Continuity of Flow in a Column

Consider a column of lubricant as shown in Figure (2.3). The lubricant flows into the column horizontally at rates of q_{x} and q_{y} and out of the column at rates of $(q_{\mathrm{x}} + \frac{\partial q_{\mathrm{x}}}{\partial x}dx)$ and $(q_{\mathrm{y}} + \frac{\partial q_{\mathrm{y}}}{\partial y}dy)$ per unit length and width, respectively. In the vertical direction the lubricant flows into the column at the rate of $w_0 dxdy$ and out of the column at the rate of $w_{\mathrm{h}} dxdy$, where w_0 is the velocity at which the bottom of the column moves up and w_0 is the velocity at which the top of the column moves up.
The principle of continuity of flow requires that the influx of a liquid must equal its efflux

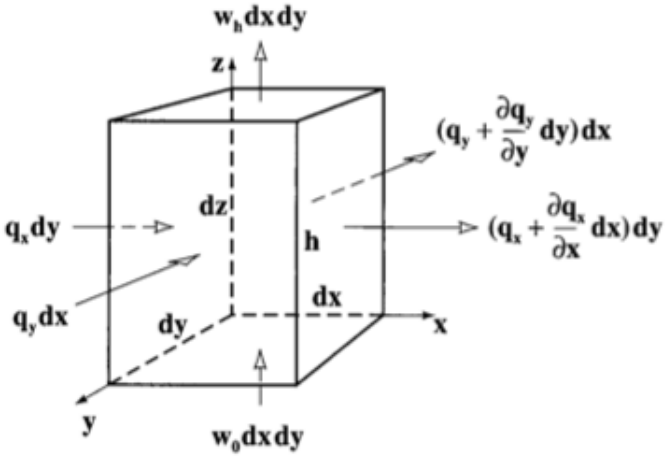


Figure 2.3: Continuity of flow in a column[7].

from a control volume under steady conditions. And with the assumption that If the density of the lubricant is constant then the following relation applies:

$$q_{\mathrm{x}}dy + q_{\mathrm{y}}dx + w_0 dxdy = \left(q_{\mathrm{x}} + \frac{\partial q_{\mathrm{x}}}{\partial x}dx\right)dy + \left(q_{\mathrm{y}} + \frac{\partial q_{\mathrm{y}}}{\partial y}dy\right)dx + w_{\mathrm{h}}dxdy \tag{2.13}$$

simplifying:

$$\frac{\partial q_{\mathrm{x}}}{\partial x}dxdy + \frac{\partial q_{\mathrm{y}}}{\partial y}dxdy + (w_{\mathrm{h}} - w_0)dxdy = 0 \tag{2.14}$$

Since $dxdy \neq 0$ equation (2.14) can be rewritten as:

$$\frac{\partial q_{\mathrm{x}}}{\partial x} + \frac{\partial q_{\mathrm{y}}}{\partial y} + (w_{\mathrm{h}} - w_0) = 0 \tag{2.15}$$

which is the equation of continuity of flow in a column.

Flow rates per unit length, q_{x} and q_{y} can be found from integrating the lubricant velocity profile over the film thickness,

$$q_{\mathrm{x}} = \int_0^h u dz \tag{2.16}$$

$$q_{\mathrm{y}} = \int_0^h v dz \tag{2.17}$$

substituting u from equation (2.11) and simplifying gives the flow rate in the 'x' direction,

$$q_{\mathrm{x}} = -\frac{h^3}{12\eta}\frac{\partial p}{\partial x} + (U_1 + U_2)\frac{h}{2} \tag{2.18}$$

Similarly as the flow rate in the 'y' direction with equation (2.12),

$$q_{\mathrm{y}} = -\frac{h^3}{12\eta}\frac{\partial p}{\partial y} + (V_1 + V_2)\frac{h}{2} \tag{2.19}$$

Substituting now for flow rates into the continuity of flow equation (2.15):

$$\frac{\partial}{\partial x}\left[-\frac{h^3}{12\eta}\frac{\partial p}{\partial x}+(U_1+U_2)\frac{h}{2}\right]+\frac{\partial}{\partial y}\left[-\frac{h^3}{12\eta}\frac{\partial p}{\partial y}+(V_1+V_2)\frac{h}{2}\right]+(w_\mathrm{h}-w_0)=0 \quad (2.20)$$

Defining $U = U_1 + U_2$ and $V = V_1 + V_2$ and assuming that there is no local variation in surface velocity in the 'x' and 'y' directions. With rearranging and simplifying yields the full Reynolds equation in three dimensions.

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right)=6\left(U\frac{dh}{dx}+V\frac{dh}{dy}\right)+12(w_\mathrm{h}-w_0) \quad (2.21)$$

2.2.3 Simplifications to the Reynolds Equation

As you can see that the Reynolds equation in (2.21) is too complex for practical engineering applications. With the following simplifications can make it more simplified.

Unidirectional Velocity Approximation

It is always possible to choose axes in such a way that one of the velocities is equal to zero, i.e., V = 0 see Figure (2.4). There are very few engineering systems, in which, for example, a journal bearing slides along a rotating shaft.

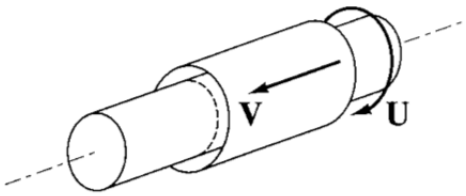


Figure 2.4: Since in there are two velocities V and U, it is possible to let one of them equals to zero. [7].

Assuming that $V = 0$ equation (2.21) can be rewritten in a more simplified form:

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right)=6U\frac{dh}{dx}+12(w_\mathrm{h}-w_0) \quad (2.22)$$

Steady Film Thickness Approximation

It is also possible to assume that there is no vertical flow across the film, i.e., $w_\mathrm{h} - w_0 = 0$. This assumption requires that the distance between the two surfaces remains constant during the operation. It is known that there is always a oil film between them.
Therefore equation (2.22) can be rewritten in the form:

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right)=6U\frac{dh}{dx} \quad (2.23)$$

Isoviscous Approximation

For many practical engineering applications it is assumed that the lubricant viscosity is constant over the film, i.e., η = constant. Then equation (2.23) can further be simplified:

$$\frac{\partial}{\partial x}\left(h^3\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(h^3\frac{\partial p}{\partial y}\right)=6U\eta\frac{dh}{dx} \tag{2.24}$$

This is in fact the most commonly quoted form of Reynolds equation throughout the literature.

Infinitely Long Bearing Approximation

The simplified Reynolds equation (2.24) is two-dimensional and numerical methods are needed to obtain a solution. Thus, further simplifying assumptions are made.
This approximation illustrated in Figure (2.5). It can be said that the pressure gradient acting along the 'y' axis which is assumed that $\partial p/\partial y = 0$ and $h \neq f(y)$. It is crucial to specify that the bearing is infinitely long in the 'y' direction. Nonetheless, the pressure gradient acting along the 'y' axis is said to be negligibly small compared to the pressure gradient acting along the 'x' axis. This assumption reduces the Reynolds equation to a one-dimensional form which is very convenient for quick engineering analysis.

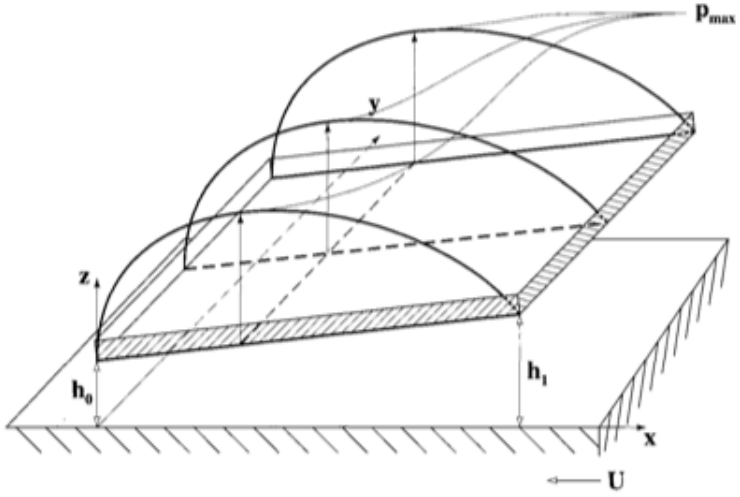


Figure 2.5: Pressure distribution in the long bearing approximation[7].

Since $\dfrac{\partial p}{\partial y}=0$, the second term of the Reynolds equation (2.24) is also zero and equation (2.24) simplifies to:

$$\frac{\partial}{\partial x}\left(h^3\frac{\partial p}{\partial x}\right)=6U\eta\frac{dh}{dx} \tag{2.25}$$

which can easily be integrated:

$$h^3\frac{\partial p}{\partial x}=6U\eta h+C \tag{2.26}$$

with the boundary condition at $h = \bar{h}$ is $dp/dx = 0$ see Figure (2.6) then substituting to (2.26) gives:

$$C = -6U\eta\bar{h}$$

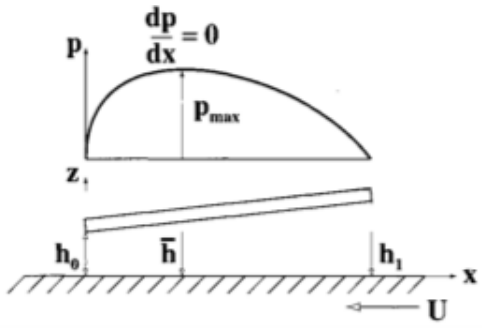


Figure 2.6: Show the boundary condition when $h = \bar{h}$ [7].

And the final form of the one-dimensional Reynolds equation for the **'long bearing approximation'** after integrating is:

$$\frac{dp}{dx} = 6U\eta\frac{h - \bar{h}}{h^3} \tag{2.27}$$

Note that the velocity 'U' in the convention assumed is negative, as shown in Figure (2.7).

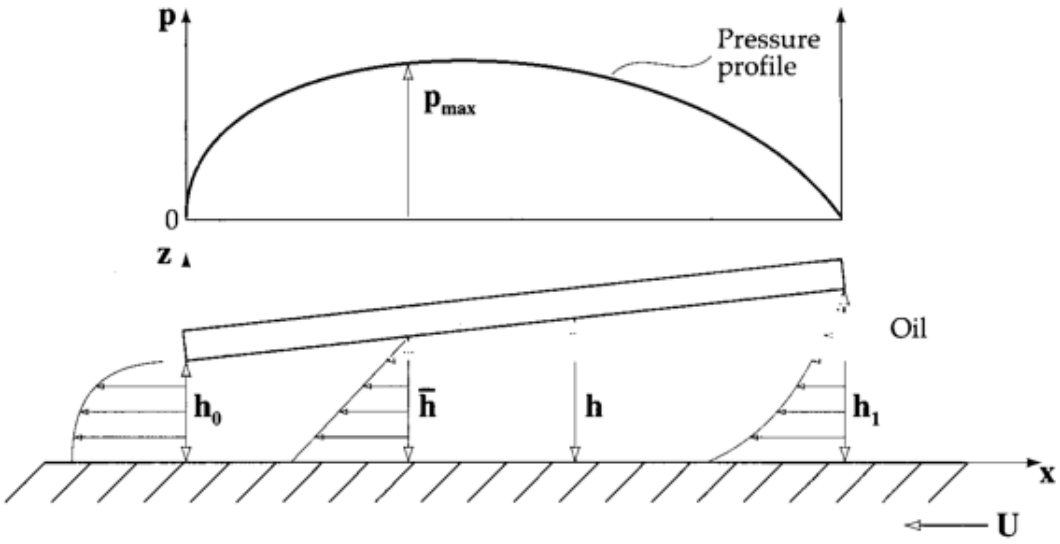


Figure 2.7: Principle of hydrodynamic pressure generation between non-parallel surfaces[7].

2.3 Bearing Parameters Predicted from Reynolds Equation of Pad Bearing

Pad bearings, which consist of a pad sliding over a smooth surface, are widely used in machinery to sustain thrust loads from shafts, e.g., from the propeller shaft in a ship shown in Figure (2.8).

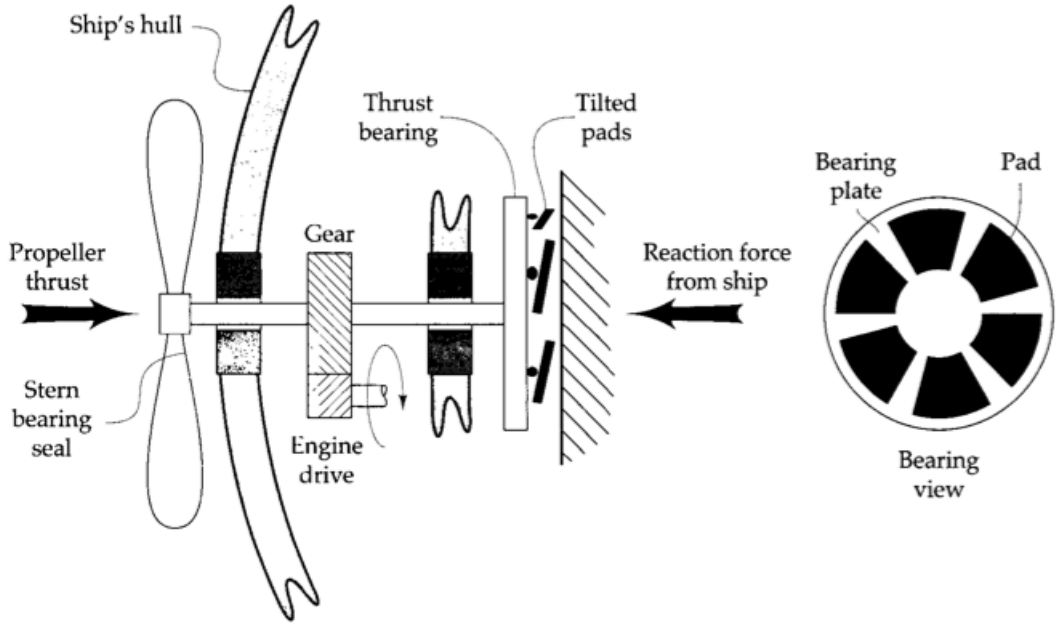


Figure 2.8: Example of a pad bearing application to sustain the thrust loads from the ship propeller shaft.[7].

The infinite linear pad bearing, as already mentioned, is a pad bearing of infinite length normal to the direction of sliding. This particular bearing geometry is the easiest to analyze. Consider an infinitely long linear wedge with L/B > 3 as shown in Figure (2.9), where 'L' and 'B' represented pad length and pad width, respectively. Both are the pad dimensions normal to and parallel to the sliding direction. Assume that the bottom surface is moving in the direction shown, dragging the lubricant into the wedge which results in pressurization of the lubricant within the wedge. The inlet and the outlet conditions of the wedge are controlled by the maximum and minimum film thicknesses, 'h_1' and 'h_0', respectively.

2.3.1 Bearing Geometry

In any bearing analysis the bearing geometry, i.e., $h = f(x)$, must be defined. The film thickness h in Figure (2.9). is expressed as:

$$h = h_0 + x \tan \alpha = h_0 + x \frac{h_1 - h_0}{B}$$

$$h = h_0 \left(1 + \frac{h_1 - h_0}{h_0} \frac{x}{B} \right)$$

Let K which is often known in the literature as 'the convergence ratio':

$$K = \frac{h_1 - h_0}{h_0}$$

The film geometry can then be expressed as:

$$h = h_0 \left(1 + \frac{Kx}{B}\right) \tag{2.28}$$

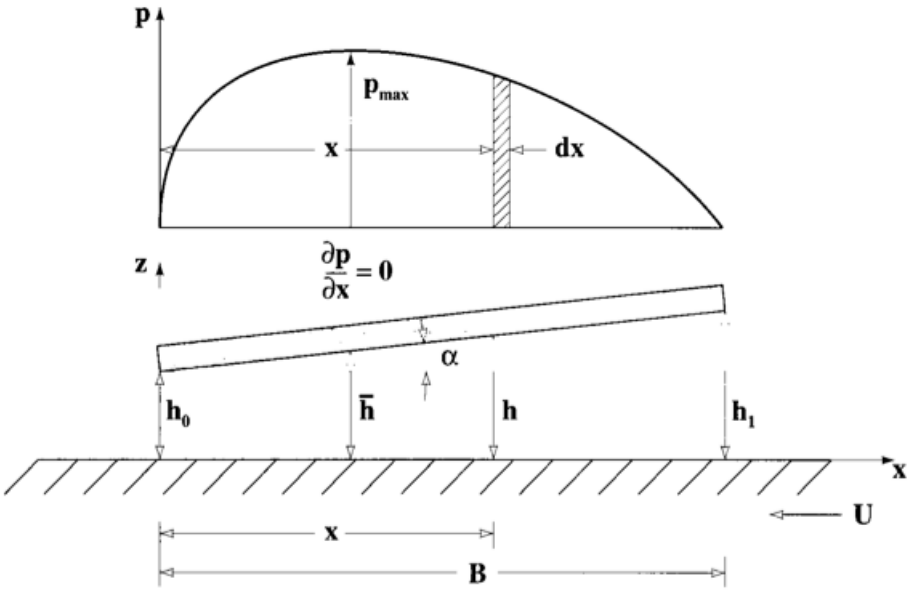


Figure 2.9: Geometry of a linear pad bearing[7].

2.3.2 Pressure Distribution

The pressure distribution can be calculated by integrating the Reynolds equation over the specific film geometry. Since the pressure gradient in the 'x' direction is dominant, the one-dimensional Reynolds equation for the long bearing approximation (2.27) can be used for the analysis of this bearing. Any by differentiating (2.28) which gives dx in terms of dh:

$$dx = \frac{B}{kh_0} dh \tag{2.29}$$

Substituting (2.29) into (2.27) and after simplifying obtains:

$$\frac{kh_0}{6U\eta B} dp = \frac{h - \bar{h}}{h^3} dh \tag{2.30}$$

which is the differential formula for pressure distribution in this bearing. Equation (2.30) can be integrated to give:

$$\frac{kh_0}{6U\eta B}p = -\frac{1}{h} + \frac{\bar{h}}{2h^2} + C \tag{2.31}$$

With the boundary conditions according to Figure (2.9):

$$\begin{aligned} p &= 0 \qquad at \qquad h = h_0 \\ p &= 0 \qquad at \qquad h = h_1 \end{aligned}$$

Substituting into (2.31) the constants '$\bar{h}$' and 'C' are:

$$\bar{h} = \frac{2h_0h_1}{h_1 + h_0}$$
$$C = \frac{1}{h_1 + h_0}$$

From the convergence ratio 'K', the maximum film thickness 'h_1' can be expressed in:

$$h_1 = h_0(K + 1) \tag{2.32}$$

Substituting into the constants '$\bar{h}$' and 'C' in terms of 'K':

$$\bar{h} = 2h_0\frac{K + 1}{K + 2}$$
$$C = \frac{1}{h_0(K + 2)}$$

Finally, substituting into equation (2.31) gives:

$$p = \frac{6U\eta B}{kh_0}\left(-\frac{1}{h} + \frac{h_0}{h^2}\frac{(K + 1)}{(K + 2)} + \frac{1}{h_0(K + 2)}\right) \tag{2.33}$$

Note that the velocity 'U' in the convention assumed is negative, as shown in Figure 2.7.

2.3.3 Load Capacity

The total load that a bearing will support at a specific film geometry is obtained by integrating the pressure distribution over the specific bearing area. If the load is varied then the film geometry will change to re-equilibrate the load and pressure field. The load that the bearing will support at a particular film geometry is:

$$W = \int_0^L \int_0^B p dx dy$$

This can be re-written in terms of load per unit length:

$$\frac{W}{L} = \int_0^B p dx$$

and substituting for p from equation (2.33) yields:

$$\frac{W}{L} = \frac{6U\eta B}{Kh_0}\int_0^B \left(-\frac{1}{h} + \frac{h_0}{h^2}\frac{(K+1)}{(K+2)} + \frac{1}{h_0(K+2)}\right) dx \tag{2.34}$$

And again substituting (2.29) for dx and integrating yields:

$$\frac{W}{L} = \frac{6U\eta B^2}{K^2{h_0}^2}\left(-ln(K+1) + \frac{2K}{K+2}\right) \tag{2.35}$$

Equation (2.35) is the total load per unit length the bearing will support expressed in terms of the bearing's geometrical and operating parameters which can be optimized to give the best performance.
Note that the velocity 'U' in the convention assumed is negative, as shown in Figure 2.7.

2.3.4 Friction Force

The friction force generated in the bearing due to the shearing of the lubricant is obtained by integrating the shear stress τ over the bearing area

$$F = \int_0^L \int_0^B \tau dx dy$$

The friction force per unit length is:

$$\frac{F}{L} = \int_0^B \tau dx$$

where shear stress is defined in terms of dynamic viscosity and shear rate:

$$\tau = \eta\frac{du}{dz}$$

where du/dz is obtained by differentiating the velocity equation (2.11).
In the bearing considered, the bottom surface is moving while the top surface remains stationary:

$$U_1 = 0 \qquad and \qquad U_2 = U$$

thus the velocity equation (2.11) is:

$$u = \left(\frac{z^2 - zh}{2\eta}\right)\frac{\partial p}{\partial x} - U\frac{z}{h} + U$$

Differentiating gives the shear rate:

$$\frac{du}{dz} = (2z - h)\frac{1}{2\eta}\frac{dp}{dx} - \frac{U}{h} \tag{2.36}$$

and substituting yields the friction force per unit length:

$$\frac{F}{L} = \int_0^B \left[\left(z - \frac{h}{2} \right) \frac{dp}{dx} - \frac{U\eta}{h} \right] dx \tag{2.37}$$

The friction force on the lower moving surface, is greater than on the upper stationary surface. At the moving surface z = 0 (as shown in Figure (2.9), hence the acting frictional force per unit length is:

$$\frac{F}{L} = -\int_0^B \frac{h}{2}\frac{dp}{dx}dx - \int_0^B \frac{U\eta}{h}dx \tag{2.38}$$

Integrating the first part of equation (2.38) by integration by parts: So if,

$$a = \frac{h}{2} \qquad and \qquad db = \frac{dp}{dx}dx$$

then:

$$da = \frac{1}{2}dh \qquad and \qquad b = \int \frac{dp}{dx}dx = p$$

substituting:

$$-\int_0^B \frac{h}{2}\frac{dp}{dx}dx = -\left(\left| \frac{h}{2}p \right|_0^B - \int_0^B \frac{1}{2}pdh \right) \tag{2.39}$$

Since p = 0 at x = 0 and at x = B see Figure (2.9) the term $\left|\frac{h}{2}p\right|_0^B$ also equals to zero. In the remaining term variables are replaced before integration and substituting for 'dh' in equation 2.29 gives:

$$-\int_0^B \frac{h}{2}\frac{dp}{dx}dx = 0 + \int_0^B \frac{1}{2}p\frac{Kh_0}{B}dx = \frac{Kh_0}{2B}\int_0^B pdx$$

Thus the first term of equation (2.38) is:

$$-\int_0^B \frac{h}{2}\frac{dp}{dx}dx = \frac{Kh_0}{B}\frac{W}{L} \tag{2.40}$$

And integrating the second term of equation (2.38):

$$\int_0^B \frac{U\eta}{h}dx = \frac{U\eta B}{Kh_0}ln(1+K) \tag{2.41}$$

Then after substituting equation (2.40) and (2.41) into equation (2.38) the expression for friction force per unit length for a linear pad bearing is obtained:

$$\frac{F}{L} = \frac{Kh_0}{2B}\frac{W}{L} - \frac{U\eta B}{h_0 K}ln(1+k) \tag{2.42}$$

Note that calculating the friction force for the upper surface, i.e., for z = h, and subtracting from equation (2.42) yields $Kh_0W/BL = Wtan\alpha/L$. Substituting for 'W' from equation (2.35) and simplifying:

$$\frac{F}{L} = \frac{U\eta B}{h_0}\left(\frac{6}{(K+2)} - \frac{4ln(K+1)}{K}\right) \tag{2.43}$$

The bearing geometry can now be optimized to give a minimum friction force, but it is more useful to optimize the bearing to find the minimum coefficient of friction since this provides the most efficient bearing geometry for any imposed load.

2.3.5 Coefficient of Friction

By definition the coefficient of friction is expressed as a ratio of the friction and normal forces acting on the surface:

$$\mu = \frac{F}{W} = \frac{F/L}{W/L} \tag{2.44}$$

substituting for F/L and W/L and simplifying:

$$\mu = \frac{Kh_0}{B}\left[\frac{3K - 2(K+2)ln(K+1)}{6K - 3(K+2)ln(K+1)}\right] \tag{2.45}$$

Even though it was written in the Chapter 1 that this work will consider to minimize only the friction force, at the end when the optimal shape is known the value of the coefficient of friction force will be presented. There are two possible way either to use this equation (2.34) or to suddenly find the ration in equation (2.33). [1]

[1]The contents of this section including the images are all referenced from the book *"Engineering Tribology Third Edition" by STACHOWIAK, Gwidon W. a Andrew W. BATCHELOR.* which is also listed in the Bibliography section[7].

Chapter 3

Newton's Method

3.1 Newton-Raphson Formula

Newton's method or Newton–Raphson method is a root-finding algorithm which is simple and widely used to solve an equation $f(x) = 0$. Suppose $f : R^1 \to R^1$ is a differentiable function. The idea is starting with the initial guess x_0. See Figure (3.1), I find the value of $f(x_0)$ then approximate the graph of f by suitable tangents. In equation (3.1) can represent the tangent of the graph[8]. Using an approximate value x_0 obtained from the graph of f and let x_1 be the point of intersection of the x-axis and the tangent to the curve of f at x_0 see equation (3.2).

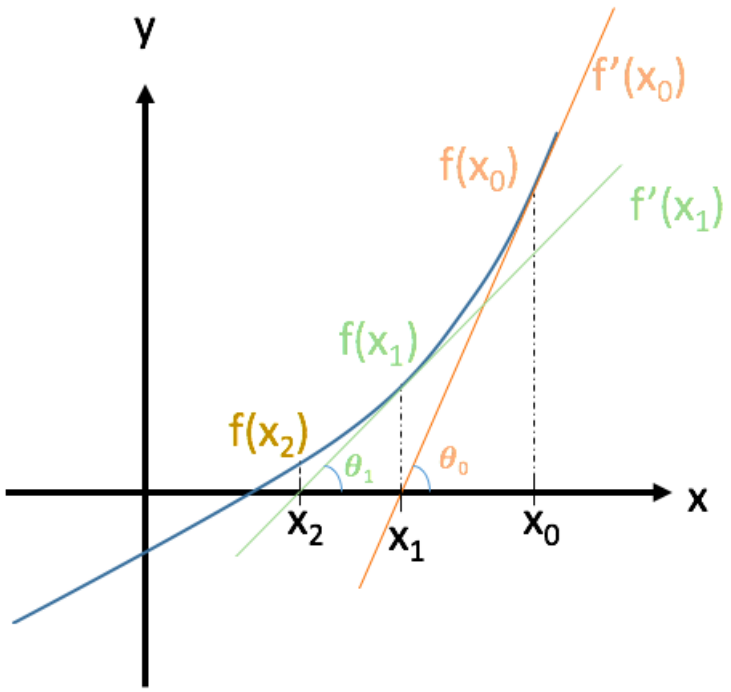


Figure 3.1: Newton–Raphson method

$$\tan\theta_0 = f'(x_0) = \frac{f(x_0)}{x_0 - x_1} \tag{3.1}$$

$$\Rightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{3.2}$$

From this the Newton-Raphson iteration formula can be written in the form:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.3}$$

3.2 Advantages and Disadvantages of Newton's method

Advantages

1. It is is one of the most powerful and well-known numerical methods for solving a root-finding problem[9].
2. When the method converges, Newton's method can provide extremely accurate approximations with very few iterations[9].
3. When comparing Newton's method to other types of functional iteration as a derive technique, it can be said that Newton's method obtains faster convergence[9].

Disadvantages

1. Newton's method does not guarantee that it will converge if the starting point or initial value is too far from the exact root and also when the tangent line becomes parallel or almost parallel to the x-axis.
2. It is clear from equation (3.3) that Newton's method cannot be continued if $f'(x_n) = 0$ for some n[9].
3. This method is computationally expensive in order to evaluate with each iteration $f(x)$ and $f'(x)$. Moreover, the term $f'(x)$ is not always possible to work with.

3.3 Jacobian in Newton's method

From the drawback of Newton's method about finding the derivative terms, therefore in this section according to Leong et al.[10] had presented that it is possible to use Jacobian while approaching Newton's method.

Chapter 4

FZERO in Matlab

4.1 History of fzero

The *fzero* command is a function file in Matlab which is developed from *zeroin* algorithm by T. Dekker in 1968[11]. Dekker's algorithm is an interesting technique combining bisection and secant method together to find a zero of a function of a real variable. After improving by R. P. Brent in 1971, *fzero* function becomes a combination of root bracketing, bisection, and inverse quadratic interpolation(IQI)[12]. It is guaranteed by Brent that this method will converge since the function can be evaluated within the initial interval. The function *fzero* can use to find numerically one solution with one variable only to the real function $f(x) = 0$. If f is continuous and changes sign in the interval $[a, b]$, with $f(a)f(b) \leqslant 0$, then there must be a root x of $f(x) = 0$ in the interval, otherwise an error message will appear.

4.2 A Combination of Root Bracketing, Bisection, and Inverse Quadratic Interpolation(IQI)

4.2.1 Inverse Quadratic Interpolation(IQI)

It is based on the idea of Lagrange interpolating polynomial in equation (6.3)[12]. The reason why it is called inverse quadratic because it needs to be swapped x's for y's then here x is interpolated as a function of $g^{(i)}$'s which intersects x-axis just once. Otherwise it would be parabola $y = x^2$ and intersects x-axis twice which is not what the method wants. And then use the equation (4.2) as the next guess. Using three points on a quadratic curve where it intersects the x-axis will obtain more accurate method.

Lagrange interpolating polynomial:

$$x = x^{(i-2)}\frac{(y-g^{(i-1)})(y-g^{(i)})}{(g^{(i-2)}-g^{(i-1)})(g^{(i-2)}-g^{(i)})} + x^{(i-1)}\frac{(y-g^{(i-2)})(y-g^{(i)})}{(g^{(i-1)}-g^{(i-2)})(g^{(i-1)}-g^{(i)})} + \\ + x^{(i)}\frac{(y-g^{(i-2)})(y-g^{(i-1)})}{(g^{(i)}-g^{(i-2)})(g^{(i)}-g^{(i-1)})} \tag{4.1}$$

The above quadratic curve intersects the x-axis while $y = 0$. Thus the next approximation is

$$x = \frac{x^{(i-2)}g^{(i-1)}g^{(i)}}{(g^{(i-2)} - g^{(i-1)})(g^{(i-2)} - g^{(i)})} + \frac{x^{(i-1)}g^{(i-2)}g^{(i)}}{(g^{(i-1)} - g^{(i-2)})(g^{(i-1)} - g^{(i)})} + \\ + \frac{x^{(i)}g^{(i-2)}g^{(i-1)}}{(g^{(i)} - g^{(i-2)})(g^{(i)} - g^{(i-1)})} \tag{4.2}$$

4.2.2 Bisection Method

Bisection method is an example of bracketing method, where two initial guess are chosen[11]. There are represented as a and b, the interval [a,b]. On which the given function $f(x)$ changes sign means $f(a) \cdot f(b) \leqslant 0$. It will start with an interval [a,b], then find the midpoint of an interval. The idea is to repeatedly cut the interval [a,b] in a half, while keeping the condition that the sign change. For example see Figure (4.1), It can be seen in the following step:

1. Starting at an interval [a,b] finds midpoint at c.
2. Considering an interval [a,c] with a midpoint d because the sign is different between point a and c not point a and b anymore
3. Finally, an interval [d,c] with $f(c) \cdot f(d) \leqslant 0$ can reach the point x where f(x) = 0.

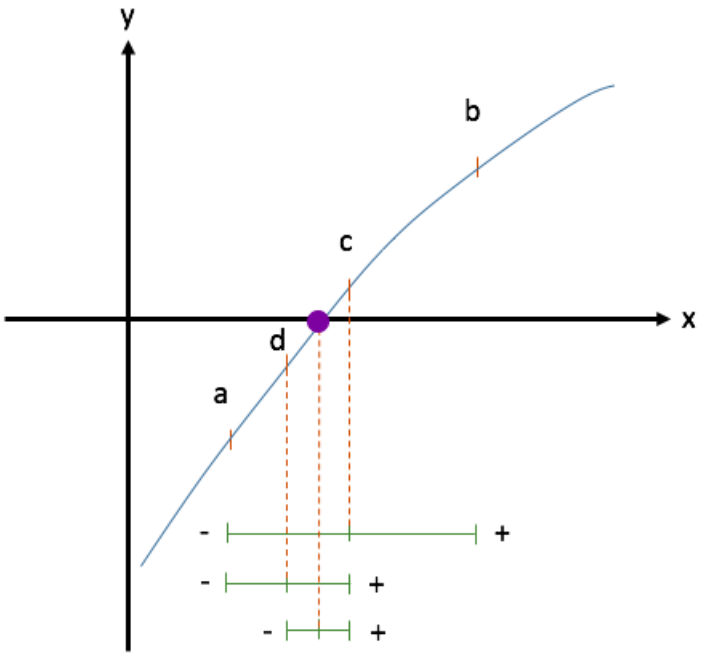


Figure 4.1: Bisection Method

Chapter 5

Genetic Algorithm

5.1 Genetic Algorithms for Optimization

Genetic Algorithms will search through a vast solution space to come up with a solution as close to the global optimum as possible based on "survival of the fittest"[13]. The method of Genetic Algorithms exploit the features of genetic evolution in nature, according to Erlend[4], he said genetic operators are utilized in order to create new individuals for the next generation in GA. The most commonly used operators are crossover-, mutation- and elite operators. Similarly, Justo[14] said there are three important steps in the algorithm selection is to generate the initial population, recombination is to evaluate the fitness function, and mutation is to generate a new population of individuals.

5.2 Matlab-based Genetic Algorithm Toolbox for Function Optimization

According to MATLAB - Optimization Toolbox - User's Guide - R2019a [15] has already described how the GA works. In Figure 5.1, the flowchart showed the simple process of GA in conclusion. At each step, the genetic algorithm selects individuals randomly from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. The Global Optimization Toolbox of MATLAB provides a complete package of genetic algorithm features which we will mainly use in later chapters. As a motivation, see the chapter 6.

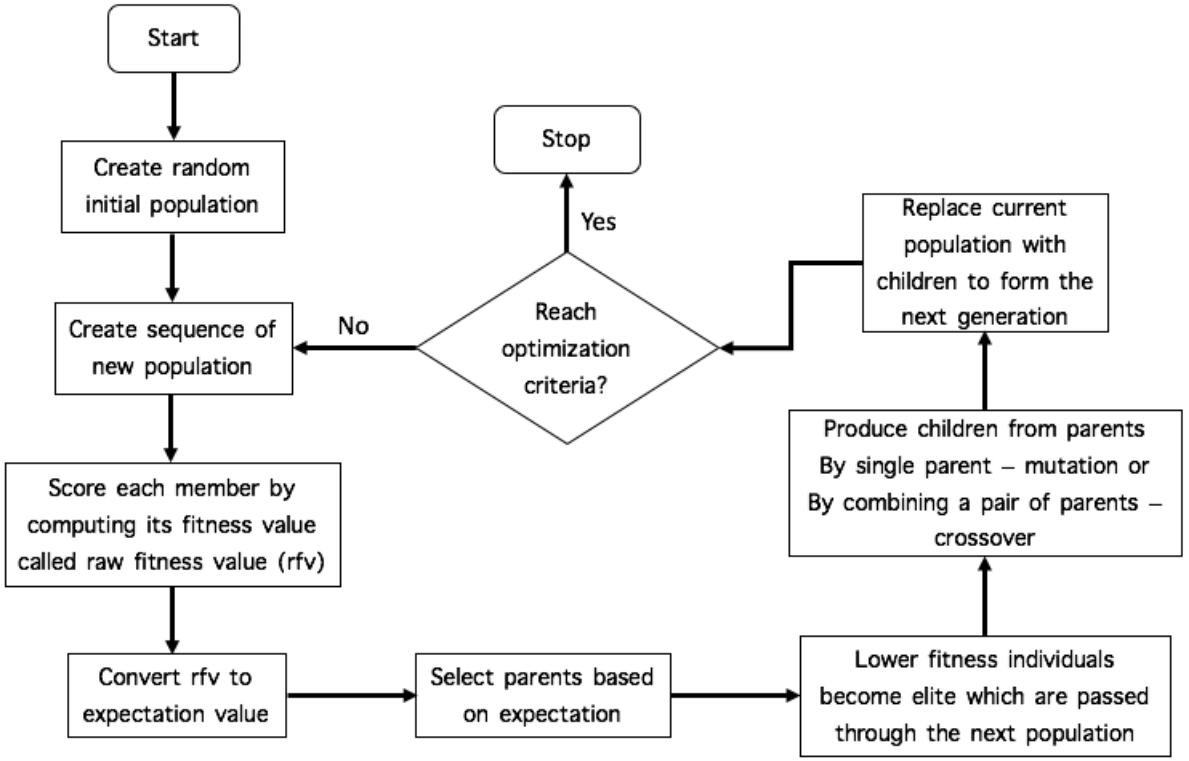


Figure 5.1: Standard procedure of simple GA (based on "GAtoolbox: a Matlab-based Genetic Algorithm Toolbox for Function Optimization")[15].

5.3 Description of GAToolbox

Justo[14] said there are four main steps to construct GA. Differently configure which is set by the user will make different optimal solution. It depends on objective function and type of variables and constraints all of these will be limitations when the user sets GA. And it will be shown in the following

1. **Problem Definition**: It is necessary to start with defining the objective function which the user would like to minimize, number of variables, constraints, and the lower bound and upper bound of each variable. It is also possible to maximize objective function by applying the property $maxf = -minf$.

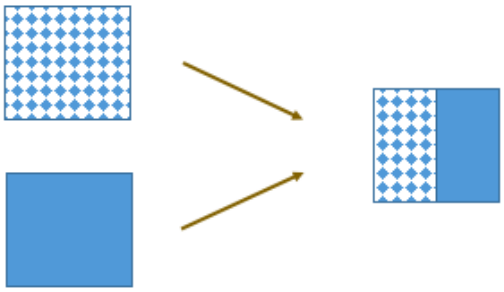
2. **Variable Setting**: It depends on the type of variable defined by the problem and the type of representation set in GA. It can be real decimal, integer decimal, and binary. It will effect how GA runs in workspace. For example, in this work also set one variable to be an integer in order to receive the number related to the type of variable.

3. **Generation of Initial Population**: First step is to create the initial population by specifying how many individuals in each generation. With a large population size, the genetic algorithm searches the solution space in a global minimum. Nevertheless, if the number has too large population size also causes the algorithm to run more slowly[15]. Moreover, if the user knows the approximate minimal point for a function lies, sets initial range so that the point lies near the middle of the range[16]. For example, using lower and upper bound of the variable.

4. **Evolution Module**: This will talk about how to crate the next generation or called children.

    - ***Selection*** At each step, GA uses the current population by selecting individuals based on expectation values as parents see Figure (5.1). An individual can be selected to be a parent more than one time, since it contributes its genes to more than one child [16]. In case having an integer variable, GA will directly select Tournament method. It will choose t numbers to be tournament size players and then they compete in tournament where the best individual out of that set will be a parent[14, 15].
    - ***Reproduction*** Since in selection section, it does not generate new individuals just only copy from parents and paste at the next generation. Reproduction options will make diversity to the population and control how GA creates the next generation. There is an option called Elite count which will allow number of individuals to survive in the next generation safely. But these value should not be high at least one is enough because it will dominate the population, which can make the search less effective[16]. Another option is Crossover fraction, specifies the fraction of the individuals and then to combine two parents to form children for the next generation called crossover children. This would be useful when GA explore for the answer.
    - ***Mutation*** In order not to get stuck in the local minimum, GA makes small random changes in the individuals in the population to create mutation children. While GA searching can go further. See Figure (5.2)

(a) Elite child



(b) Crossover child

(c) Mutation child

Figure 5.2: Schematic diagram illustrates the three types of children[16].

Note that: It is possible to reduce time when GA searching for the optimal solution by using parallel computing[4].

5.4 Advantages and Disadvantages of Genetic Algorithm

Advantages of Genetic Algorithm:

1. Instead of deterministic, GA adopts stochastic optimization to search for a solution that will avoid local optimum[17].

2. In the searching space, GA considers many points simultaneously, not only a single point, but also to deal with large parameter spaces.

3. In implementation, GA is easy to be generated.

4. In setting objective function, it does not require a continuous one. Thus, it can handle both discrete and continuous parameters.

5. In case you have a multiobjective optimization problem, you can also use GA which will compute in one run.

Disadvantages of Genetic Algorithm:

1. The problem of identifying fitness function or objective function[17].
2. The problem of choosing the various parameters such as the size of the population, mutation rate, crossover rate, the selection method and etc.
3. The problem of finding the exact global optimum because the genetic algorithm uses random number generators, the algorithm returns slightly different results each time you run it.

Chapter 6

Real Data Application

In this chapter will describe how to find the objective function and show how the methodology works. This is necessary to conclude what is the final optimal geometry of the thrust bearing and analysis of the obtained output.

6.1 Objective Function

Since the aim of this work is to find shape optimal design of thrust bearing in order to minimize the friction force. Therefore, the equation(2.43) is changed to be the objective function as the following:

$$minF = \frac{LU\eta B}{h_0}\left(\frac{6}{(K+2)} - \frac{4ln(K+1)}{K}\right) \tag{6.1}$$

6.2 Find Oil Film Thickness h_0

Oil film thickness (h_0) is a value representing small space between shaft and thrust bearing that these two things do not touch each other. And this value will be used to be a connecting variable between equation (2.35) of loading capacity (W) and equation (2.43) of friction force (F). Now it is the right time to refer to four parameters: number of segments(m), angle of running surface (α), segment inner radius (R_0), segment outer radius (R_1) considering with loading capacities and angular speeds(rpm) from Table (1.1). In addition, U in both equation is velocity so it is necessary to convert rpm[1/min] to be meter per second[m/s] with the following:

$$U[m/s] = rpm\left[\frac{1}{min}\right] \cdot 2\pi\left(\frac{R_0+R_1}{2}\right)[m] \cdot \frac{1}{60}\left[\frac{min}{sec}\right]$$

And there are few parameters for both equations:

Length of the thrust bearing:

$$L = R_1 - R_0$$

Tapered land width:

$$B = \frac{2\pi}{m}\frac{(R_0 + R_1)}{2}$$

In chapter 2, K is convergence ratio, but if we consider from Figure (2.9), it can be seen that:

$$K = \frac{Btan\alpha}{h_0}$$

It started with substituting all the known values in the equation (2.35) then find h_0 from the fact that this h_0 will make the equation equal to zero. For example, in state 1 with the values in Table (6.1)

Parameters	**Values**
m	8
α	0.3°
R_0	0.014 m
R_1	0.028 m
L	0.014 m
B	0.0165 m
W	220 N
U	0.044 m/s
η	0.01 Pa.s
K	$0.0051/h_0$

Table 6.1: Show all known parameters with example values.

Substituting in equation (2.35):

$$220 = \frac{6 \cdot 0.044 \cdot 0.01}{{h_0}^2}\left(-ln((0.0051/h_0) + 1) + \frac{2(0.0051/h_0)}{(0.0051/h_0) + 2}\right) \tag{6.2}$$

It can be seen that equation (6.2) is not a linear equation so needed some method to find the value of h_0. Next, there will be two methods, Newton's method and *fzero* command in Matlab, which are widely used to find the root of the function.

6.2.1 Newton's Method

The idea to use Newton's method which already stated in Chapter 3 needs the initial point to start. This method is applied in Matlab. The initial guess will be the value around 0.0005. The reason why this value is small because do not forget that h_0 is represented

the small space just only oil can flow through this area. Even though it is possible to set the initial guess, it cannot be sure that this is a good number. For the next iteration in equation (3.3), it would be quite difficult and complicated to find the derivative terms of equation (2.35). However, it is possible to apply Jacobian to assist Newton's method at this point. As you can see even though there are many drawbacks for using Newton's method, eventually it is always come up with an idea to solve them.

Applying Jacobian to Newton's Method

In equation (3.3) Newton's method, it is necessary to find the derivative terms but in the following idea would like to represent this term by using Jacobian instead.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Considering equation (3.1), it is said about tangent value therefore see figure (6.1)

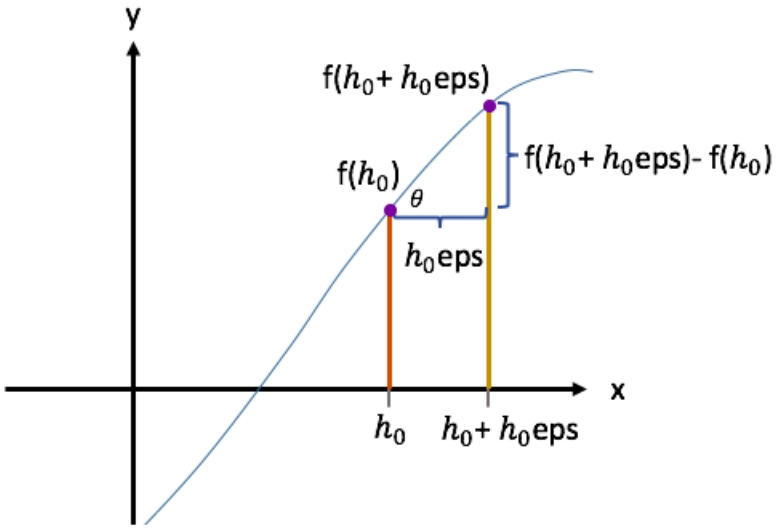


Figure 6.1: Newton's method with Jacobian.

It is evident from the Figure (6.1) that:

$$tan\theta = \frac{f(h_0 + h_0 eps) - f(h_0)}{h_0 eps} \tag{6.3}$$

The computational model is shown in Appendix A which is come from the following:

1. It is started to compute with an initial guess at $h_0 = 0.0005$, then compute K and substitute in equation (2.35) for finding the value of loading capacity(W) with this initial h_0.

2. Find the different between computed W with h_0 which is $f(h_0)$ and determined W from Table (1.1). $f = f(h_0) - W$.

3. It is given a number Δx in this case is $h_0 eps$, with this number finding another point and do a calculation as mentioned above. $ff = f(h_0 + eps) - W$.

4. Let Jacobian equals to $(ff - f)/h_0 eps$ at the same this number is also a $tan\theta$ from the Figure (6.1)

5. For the next iteration of h_0 is computed by $h_0 = h_0 - 0.0009 \cdot f/jacobian$.

6. Comparing f and W by making an ratio to know the error of this h_0. It will compute until the error from doing iteration is less than $1e-4$, i.e., the approximation point is closed or almost equal to the root answer.

For the above example, this Newton's method can give the proper value of h_0 equal to $3.1483e-05$. But if there is some change such as $\alpha = 0.1$, h_0 will be equal to $3.4334e-05+1.3443e-13i$, which is a complex number. The problem of this method might come from the convergence of the iteration since h_0 is very small and close to zero then sometimes it will go to a negative value and since in the equation there is a natural logarithm term so that it gave complex number. To avoid negative values can possibly do, but it cannot give the value of the answer since there is an acceptable error. And one more consideration point, for the next iteration of h_0 as mentioned before and can be seen clearly in the equation of Matlab file in Listing 2:

$$h_0 = h_0 - 0.0009 \cdot f/jacobian$$

The coefficient of the term $f/jacobian$ is too small, which is not normal for using Newton's method due to the claim of this method is fast and simple. From all above reasons can conclude that Newton's method does not work properly in this situation.

6.2.2 *fzero* Command in Matlab

Since Matlab has already provided the command to find a root of nonlinear function, *fzero*. So that we would say this is the most suitable way to find the root answer of h_0. To construct this command, it is important to concern the form of the function which is needed to find the root. For the computational code can be found in Appendix A. Walter, who is the professional in using Matlab suggested that first of all it should be started with considering variable h_0 in Listing (3) after arranging the equation (6.2), it is found that this equation has a quadratic form:

$$constant_1 - constant_2/{h_0}^2 = 0 \tag{6.4}$$

then multiplying through by ${h_0}^2$:

$$constant_1 \cdot {h_0}^2 - constant_2 = 0 \tag{6.5}$$

which has solutions:

$$h_0 = +/- \sqrt{constant/anotherconstant} \tag{6.6}$$

There will be two answer of this solution since It is quadratic. It means that it is necessary to provide the bounds of the solution in order to avoid unwanted answer. According to the value of h_0, one side of the bounds should be real minimum number and another is a finite number which is big enough to receive correct value.

fzero command in Matlab to find the result of h_0 is running so fast and the error can be acceptable. From equation (6.2) *fzero* also gave h_0 equal to $3.1481e-05$ which is the same as Newton's method. Therefore from now on *fzero* would be the applied method to find h_0 in this project.

6.3 Find Optimal Geometry of Thrust Bearing

There would be two algorithms to compare an optimal geometry of thrust bearing between Brute force algorithm and Genetic algorithm

6.3.1 Brute Force Algorithm

This algorithm is used to clarify what is the step by step of finding optimal shape of thrust bearing in thirteen states. The procedure is the following:

1. Begin with dividing the range of each four parameters m, α, R_0, and R_1 from scope of work in Chapter 1, into ten partitions. In this step would have all the possible number of each four parameters.

2. Form a group of four parameters from previous step as an array to substitute in equation (2.35), but do not forget to partition m only in an integer number. The aim of this step is when substituting can find the value of h_0 by *fzero* command.

3. Recheck the value of loading capacity (W), Is it still equal to a number when it is an input. Then can compute the friction force and there would be $10 \cdot 10 \cdot 10 \cdot 10 = 10,000$ possibilities to compute friction force in one state.

However, it is widely known that using Brute force algorithm will be time consuming and the partition might not be the great value to obtain the optimal value for each parameter. But brute force algorithm is helpful because it is clearly understandable when trying to figure the code in Matlab out. See the result in Table (6.2). The details of this algorithm can be seen in Appendix B.

State	W**[N]**	**rpm[1/min]**	m	α [°]	R_0**[mm]**	R_1**[mm]**	F**[N]**
1	220	20000	9	0.100	0.015	0.020	1.1643
2	256	22500	9	0.100	0.015	0.020	1.3303
3	343	25000	4	0.100	0.015	0.020	1.6134
4	462	27500	8	0.100	0.015	0.020	1.9568
5	567	30000	7	0.100	0.015	0.020	2.2629
6	657	32500	7	0.100	0.015	0.020	2.5359
7	772	35000	12	0.100	0.015	0.020	2.8550
8	899	37500	4	0.100	0.015	0.020	3.1936
9	1012	40000	10	0.100	0.015	0.020	3.5039
10	1059	42500	5	0.100	0.015	0.020	3.6932
11	994	45000	6	0.100	0.015	0.020	3.6734
12	537	47500	9	0.100	0.015	0.020	2.8002
13	100	50000	10	0.280	0.015	0.020	1.2286

Table 6.2: The result from running 13 states by brute force algorithm.

6.3.2 Genetic Algorithm(GA)

Fortunately, the Global Optimization Toolbox of Matlab has already created a complete package of genetic algorithm features. It is very easy to open an Optimization Toolbox using solver GA and input data directly see Figure (6.2). To be more effective by setting ga syntax in Matlab function, for example, $x = ga(fun, nvars, A, b, [], [], lb, ub, nonlcon, Int-Con, options)$ [16]. GA is an algorithm allowed the parameter to be an integer. According to this work, there is one parameter, number of segments (m) needed to be set as an integer. How to construct GA in Matlab based on instructions from Chapter 5 the can find below and the code in Matlab can find in Appendix B:

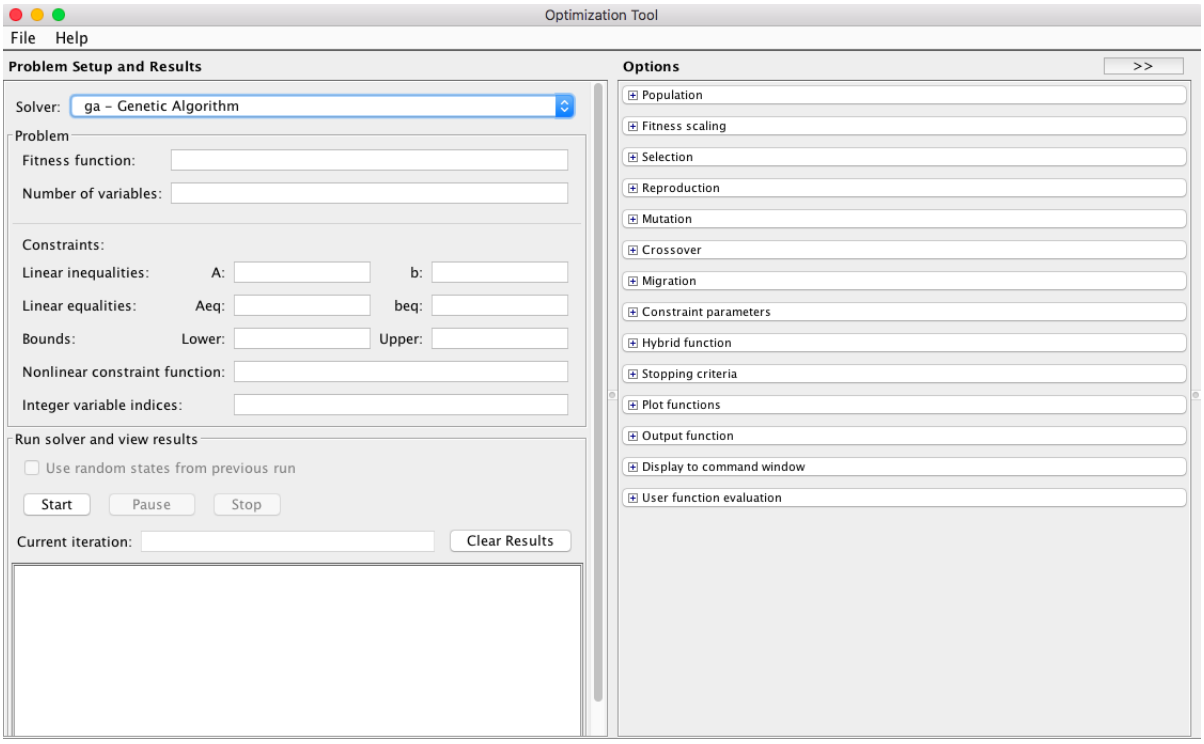


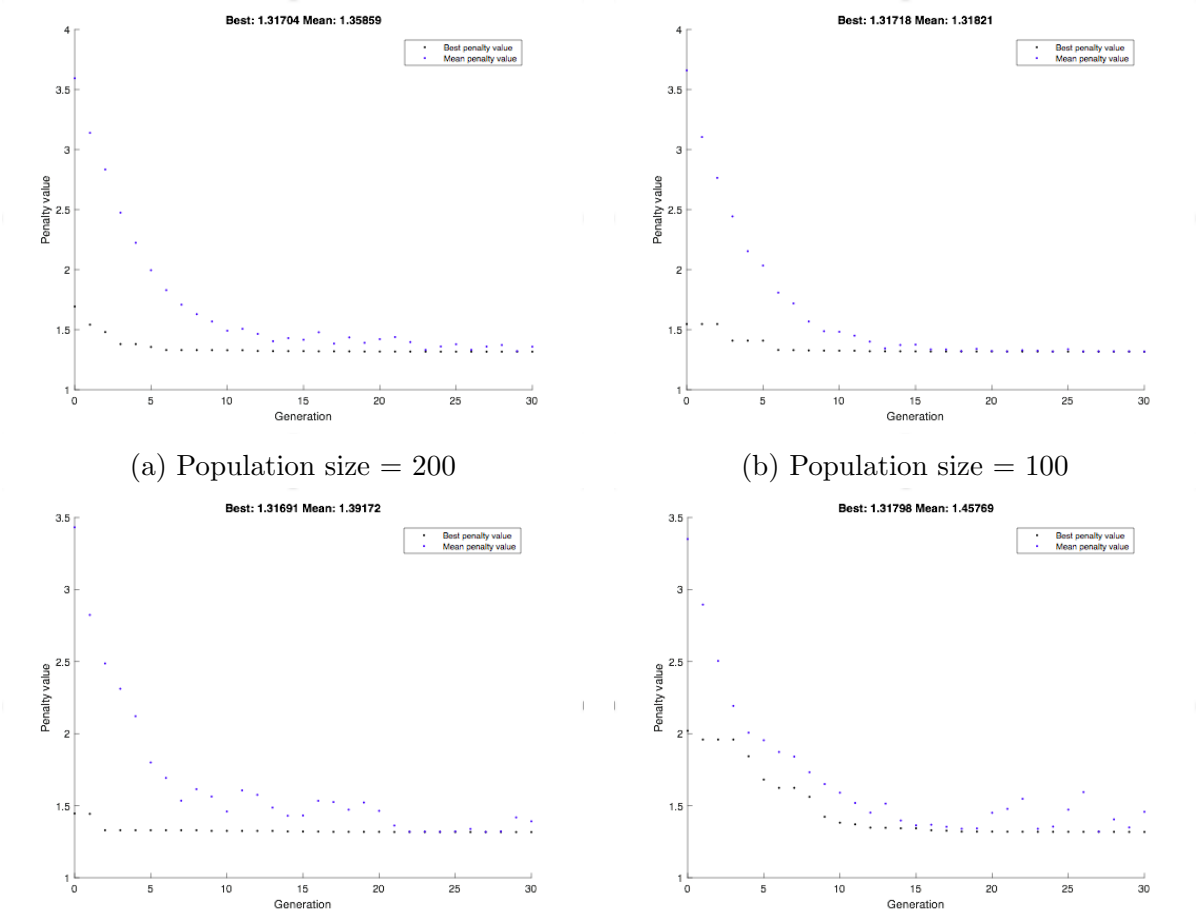
Figure 6.2: Represent the GA Toolbox in Matlab program

1. GA is set to have objective function to minimize the friction force in equation (6.1) in order to find an optimal shape design of thrust bearing.

2. Set the number of variables equal to four. Then define the range of each variable so that can help GA to find the solution in "Population Initial Range" [16]

3. Generate all the options that will be used for GA searching. Since having an integer variable there are some restrictions such as only 'doubleVector' population type, only the binary 'tournament' in selection function option [15].

4. About setting population size [15] is written that if this number is big can reach global optimum but it might take time for searching. Accordingly, in Figure (6.3) illustrates that when population size is 200 obtaining the friction force same as the population size is 25. And it is ended up with a number that population size can be equal to 25.



(a) Population size = 200

(b) Population size = 100

(c) Population size = 50

(d) Population size = 25

Figure 6.3: Plots of population size 200, 100, 50, and 25, respectively with 30 generations and loading capacity at 256 N. They represented that there is no difference to set population size between 200 and 25 individuals. We will obtain closed friction force values.

5. It can be set to use 'parallel' computing as always to reduce time calculation[4].

This algorithm can search through all possible values of four parameters and can give lower friction force and spend less time comparing to Brute force algorithm. And one drawback for GA within the code mentioned in Appendix B can calculate just only one state per time running so it means that to receive thirteen state needed to run thirteen times. The results from GA is in Table (6.3).

State	W**[N]**	**rpm[1/min]**	m	α [°]	R_0**[mm]**	R_1**[mm]**	F**[N]**
1	220	20000	9	0.1230	0.015	0.020	1.1525
2	256	22500	3	0.1142	0.015	0.020	1.3207
3	343	25000	9	0.1130	0.015	0.020	1.6075
4	462	27500	3	0.1002	0.015	0.020	1.9568
5	567	30000	3	0.1000	0.015	0.020	2.2629
6	657	32500	6	0.1000	0.015	0.020	2.5359
7	772	35000	12	0.1000	0.015	0.020	2.8550
8	899	37500	8	0.1000	0.015	0.020	3.1937
9	1012	40000	5	0.1000	0.015	0.020	3.5039
10	1059	42500	4	0.1000	0.015	0.020	3.6935
11	994	45000	6	0.1000	0.015	0.020	3.6734
12	537	47500	12	0.1223	0.015	0.020	2.7733
13	100	50000	6	0.3073	0.015	0.020	1.2323

Table 6.3: The result from running 13 states by GA.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

During running GA, it is found that the parameter m which is the number of segments does not have an influence on objective function to minimize friction force see Table (7.1). It might happen because in this project considered the part just only in one segment. This might be the reason why obtained this result. There are computations to support this conclusion. They are set to have the same values of W, rpm, α, R_0, and R_1. The values of parameter m are different between 3 and 12 followed the range of these m. All of these parameters are substituted in equation (2.35) to compute h_0 first and then into equation (2.43) finding friction force similar to equation (6.2). With these values of all parameters of all different ten numbers of m, the results in Table (7.1) illustrate that m does not effect the value of friction force even though it gives the different values of oil film thickness(h_0). Moreover, it can be seen that when the loading capacity is getting higher, the friction force is also increasing same as the assumption said in Chapter 1.

7.1.1 Friction Power Loss

And according to thirteen states which have mentioned at the beginning, here will state how to conclude the final optimal shape of this thrust bearing. Because in real situation, it is impossible to always change the thrust bearing every time the loading capacity changed. Thus, this will use the friction power loss equation:

$$P_{z_i} = F_i \cdot R_{1i} \cdot \omega_i \tag{7.1}$$

Where P_{z_i} is friction power loss at i state, F_i is friction force at i state, and ω is angular speed (rps) at i state. Friction power loss can apply to this conclusion due to every variable in the equation (7.1) is known and still be in the case of regarding to friction force. Furthermore, to be more precise by including weight function to this equation with the fact that the summation of weight function is equal to one. Additionally, combine the above ideas together with a goal to minimize friction power loss to conclude the optimal thrust bearing. Then the equation(7.1) would be changed to:

$$minP_z = \sum_{i=1}^{13} F_i \cdot R_{1i} \cdot \omega_i \cdot w_i \tag{7.2}$$

Where w_i is a weight function at state i.

Number	W[**N**]	**rpm[1/min]**	m	α [°]	R_0[**mm**]	R_1[**mm**]	h_0 [**m**]	F[**N**]
1	220	20000	3	0.1000	0.015	0.020	4.20E-05	1.1643
2	220	20000	4	0.1000	0.015	0.020	3.15E-05	1.1643
3	220	20000	5	0.1000	0.015	0.020	2.52E-05	1.1643
4	220	20000	6	0.1000	0.015	0.020	2.10E-05	1.1643
5	220	20000	7	0.1000	0.015	0.020	1.80E-05	1.1643
6	220	20000	8	0.1000	0.015	0.020	1.57E-05	1.1643
7	220	20000	9	0.1000	0.015	0.020	1.40E-05	1.1643
8	220	20000	10	0.1000	0.015	0.020	1.26E-05	1.1643
9	220	20000	11	0.1000	0.015	0.020	1.14E-05	1.1643
10	220	20000	12	0.1000	0.015	0.020	1.05E-05	1.1643

Table 7.1: Shows the computation receives the same values of friction force from different values of parameter m with the same values of others parameters

7.1.2 Weight Function

Now it is the right time to change the objective function in GA from minimize friction force to minimize friction power loss instead in order to conclude all thirteen states and at the same time reached minimum value of friction force in each state as an outcome. Owing to the equation (6.2), if it is considered in different state, it is essential to define the parameters to be unique see Table (7.3). Because in equation (2.43) are required to make them different. With careful observation since they are in different state which will effect to value of parameters. For example, a computation of h_0 with *fzero* is changed related to state changing, then the value of h_0 will effected the value of K. Therefore, in order to have smoothly calculation and to avoid any mistakes, each state is needed to have its own equation including its own variables. These idea will be used to compute in Appendix C. And the following will state the two possible ideas to generate the weight function:

1. It is assumed that every state could work at the same frequent rate see Table(7.2) since there is no information about the working process in reality.

State	W **[N]**	**rpm [1/min]**	**Weight Function**
1	220	20000	1/13
2	256	22500	1/13
3	343	25000	1/13
4	462	27500	1/13
5	567	30000	1/13
6	657	32500	1/13
7	772	35000	1/13
8	899	37500	1/13
9	1012	40000	1/13
10	1059	42500	1/13
11	994	45000	1/13
12	537	47500	1/13
13	100	50000	1/13

Table 7.2: Generate the weight function for each state when there is no information about possibility.

State	W**[N]**	**rpm[1/min]**	m	α [°]	R_0**[mm]**	R_1**[mm]**	h_0 **[m]**	**K**	F**[N]**
1	220	20000	m_1	$alpha_1$	R_{01}	R_{11}	h_{01}	K_1	F_1
2	256	22500	m_2	$alpha_2$	R_{02}	R_{12}	h_{02}	K_2	F_2
3	343	25000	m_3	$alpha_3$	R_{03}	R_{13}	h_{03}	K_3	F_3
4	462	27500	m_4	$alpha_4$	R_{04}	R_{14}	h_{04}	K_4	F_4
5	567	30000	m_5	$alpha_5$	R_{05}	R_{15}	h_{05}	K_5	F_5
6	657	32500	m_6	$alpha_6$	R_{06}	R_{16}	h_{06}	K_6	F_6
7	772	35000	m_7	$alpha_7$	R_{07}	R_{17}	h_{07}	K_7	F_7
8	899	37500	m_8	$alpha_8$	R_{08}	R_{18}	h_{08}	K_8	F_8
9	1012	40000	m_9	$alpha_9$	R_{09}	R_{19}	h_{09}	K_9	F_9
10	1059	42500	m_{10}	$alpha_{10}$	R_{010}	R_{110}	h_{010}	K_{10}	F_{10}
11	994	45000	m_{11}	$alpha_{11}$	R_{011}	R_{111}	h_{011}	K_{11}	F_{11}
12	537	47500	m_{12}	$alpha_{12}$	R_{012}	R_{112}	h_{012}	K_{12}	F_{12}
13	100	50000	m_{13}	$alpha_{13}$	R_{013}	R_{113}	h_{013}	K_{13}	F_{13}

Table 7.3: Represents the parameters used to compute the friction power loss in GA

After substituting weight function from Table (7.2) into equation (7.2) the result is in Table (7.4) and can find the computational code in Appendix C.

Parameters	**Values**
α	0.100
R_0	0.015
R_1	0.020
P_z	30.378

Table 7.4: Table of the result of equation(7.2) from the weight function at Table(7.1).

2. According to Assoc. prof. MSc. Pavel Novotný, Ph.D's experience, thrust bearing considered in this project is usually worked at angular speed around $42,500-45,000$ [1/min] and most of the time it is rarely to have an angular speed around state 1 to state 8. Thus, within this fact can generate the weight function in Table (7.5).

State	W **[N]**	**rpm [1/min]**	**Weight Function**
1	220	20000	1/145
2	256	22500	1/145
3	343	25000	1/145
4	462	27500	1/145
5	567	30000	1/145
6	657	32500	1/145
7	772	35000	1/145
8	899	37500	1/145
9	1012	40000	69/580
10	1059	42500	7/20
11	994	45000	7/20
12	537	47500	69/580
13	100	50000	1/145

Table 7.5: Generate the weight function for each state according to reality. At state 10 and 11 are said to be the most frequent to work with thrust bearing

After substituting this weight function based on specialist's experience to equation (7.2) the result is in Table (7.6) and for the computational code can generate easily

from substituting the weight function in Table(7.5) into w of Listing 7.6 in Appendix C instead of weight function in Table (7.2).

Parameters	Values
α	0.100
R_0	0.015
R_1	0.020
P_z	49.786

Table 7.6: Table of the result of equation(7.2) from the weight function at Table(7.3).

As the result in Listing (7.1) shows the influence of m, so the value of m is set to be equal to 7 as an average value of the fact that number of segments is around 4 to 10 which is known from Assoc. prof. MSc. Pavel Novotný, Ph.D's experience. Finally, the result from Table (7.4) and Table (7.6) are going to the same result that $\alpha = 0.1$ °, $R_0 = 15$ mm, and $R_1 = 20$ mm see Figure (7.1). The Table (7.7) illustrated the value of friction coefficient after applying the values of parameters mentioned above. And there is one more column represented the value of friction coefficient (μ) which is said in the Chapter 2 that it would be more useful to minimize coefficient of friction[7]. In this computation to find μ, it is computed with both two equations (2.44) and equation (2.45). Obviously, they are equal. The comparison is happened to just only make sure that the whole computation about to find the oil film thickness(h_0) and the friction force is totally correct and follow the truth of equation (2.45).

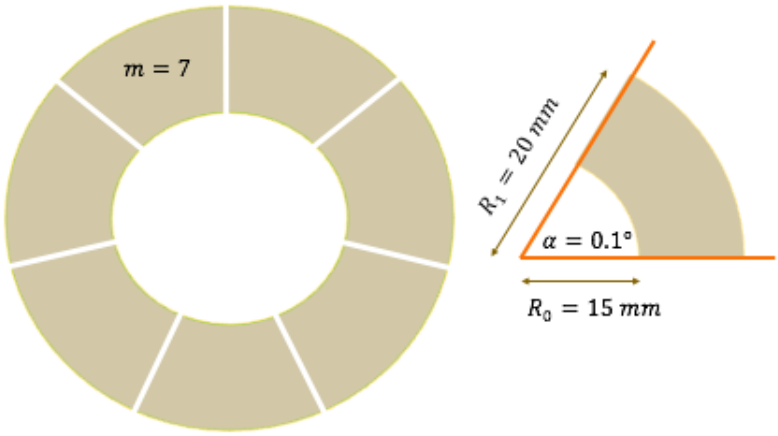


Figure 7.1: The final shape optimal thrust bearing for thirteen states.

The code for all computations enclosed in appendices are included on the CD. we hope this thesis will be useful for anyone, who is interested in shape optimization by using a great advantage of computer and algorithms.

State	W**[N]**	**rpm[1/min]**	m	α[°]	R_0**[mm]**	R_1**[mm]**	**F[N]**	μ
1	220	20000	7	0.1	0.015	0.020	1.1643	0.00529
2	256	22500	7	0.1	0.015	0.020	1.3303	0.00520
3	343	25000	7	0.1	0.015	0.020	1.6134	0.00470
4	462	27500	7	0.1	0.015	0.020	1.9568	0.00424
5	567	30000	7	0.1	0.015	0.020	2.2629	0.00399
6	657	32500	7	0.1	0.015	0.020	2.5359	0.00386
7	772	35000	7	0.1	0.015	0.020	2.8550	0.00370
8	899	37500	7	0.1	0.015	0.020	3.1936	0.00355
9	1012	40000	7	0.1	0.015	0.020	3.5039	0.00346
10	1059	42500	7	0.1	0.015	0.020	3.6932	0.00349
11	994	45000	7	0.1	0.015	0.020	3.6734	0.00370
12	537	47500	7	0.1	0.015	0.020	2.8002	0.00521
13	100	50000	7	0.1	0.015	0.020	1.4388	0.01439

Table 7.7: Values of friction coefficient (μ) after applying final shape optimization of thrust bearing.

7.2 Future Work

1. As mentioned above about the weight function to conclude the shape optimization, if someone can get the statistic data of frequency, will come up with the real one to put in the equation(7.2).

2. If there were more parameters or more constraints about generating thrust bearing, it would be more powerful to conclude shape optimization. For example, the diameter of the shaft can help us know the value of R_0.

Bibliography

[1] LEE In-Beom, HONG Seong-Ki, and CHOI Bok-Lok. Investigation of The Axial Thrust Load Using Numerical and Experimental Techniques during Turbocharger Operation. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 232(6):755–765, 2018.

[2] CHARITOPOULOS Anastassios, VISSER Roel, ELING Rob, and PAPADOPOULOS Christos. Design Optimization of an Automotive Turbocharger Thrust Bearing Using a CFD-based THD Computational Approach. *Lubricants*, 6(1):21, 2018.

[3] FAROOQ AHMAD Najar and G.A. Harmain. Numerical Investigation of Pressure Profile in Hydrodynamic Lubrication Thrust Bearing. *International scholarly research notices*, 2014, 2014.

[4] ENGEVIK Erlend L. Optimal Design of Tidal Power Generator Using Stochastic Optimization Techniques. Master's thesis, Institutt for Elkraftteknikk, 2014.

[5] SJÖBERG Erik. Friction Characterization of Turbocharger Bearings. *Master of Science Thesis, KTH Royal Institute of Technology, Stockholm, Sweden*, 2013.

[6] NGUYEN-SCHÄFER Hung. *Rotordynamics of Automotive Turbochargers*. Ludwigsburg, Germany: Springer, 2015. ISBN 978-3-319-17643-7.

[7] STACHOWIAK Gwidon W. and BATCHELOR Andrew W. *Engineering Tribology.3.* Butterworth-Heinemann, 2005. ISBN 0-7506-7836-4.

[8] RUKMANGADACHARI E. *Mathematical Methods*. Pearson Education India, 2009. ISBN 978-8-131-72598-6.

[9] BURDEN Richard L. and FAIRES J. Douglas. *Numerical Analysis*. Brooks/Cole, 2011. ISBN: 978-0-538-73351-9.

[10] WAZIRI M.Y., LEONG W.J., HASSAN M.A., and MONSI M. Jacobian Computation-Free Newton Method for Systems of Non-Linear Equations. *Journal of numerical Mathematics and stochastic*, 2(1):54–63, 2010.

[11] MOLER Cleve. Zeroin, Part 1: Dekker's Algorithm. `https://blogs.mathworks.com/cleve/2015/10/12/zeroin-part-1-dekkers-algorithm/`, 2015.

[12] PRESS William H., TEUKOLSKY Saul A., VETTERLING William T., and FLANNERY Brian P. *Numerical Recipes in FORTRAN 77 The Art of Scientific Computing.2.*, volume 1. Cambridge England, 1996. ISBN 0-521-43064-X.

[13] NALBANDH A.H. and RAJYAGURU C.C. Fixture Design Optimization Using Genetic Algorithm-a Review. *Journal of information, knowledge and research in mechanical engineering*, 2:466–471, 2013.

[14] ROBERTS J.J., CASSULA Agnelo Marotta, SILVEIRA José Luz, PRADO Pedro Osvaldo, and JUNIOR José Celso Freire. Gatoolbox: a Matlab-based Genetic Algorithm Toolbox for Function Optimization. In *THE 12TH LATIN-AMERICAN CONGRESS ON ELECTRICITY GENERATION AND TRANSMISSION-CLAGTEE*, volume 12, page 2017, 2017.

[15] Matlab Team. Matlab - Optimization Toolbox - User's Guide - R2019a. `https://www.mathworks.com/help/pdf_doc/gads/gads_tb.pdf`, 2019.

[16] Matlab Team. How the Genetic Algorithm Works. `https://ch.mathworks.com/help/gads/how-the-genetic-algorithm-works.html`, 2019.

[17] LI Tiejun, SHAO Guifang, ZUO Wangda, and HUANG Sen. Genetic Algorithm for Building Optimization: State-of-the-Art Survey. In *Proceedings of the 9th International Conference on Machine Learning and Computing*, pages 205–210. ACM, 2017.

List of Figures

List of Tables

Appendix A

Listing 1: This is a Matlab file for finding h_0 by Newton's method.

```
function h0 = GETH0byNewton

h0eps = 0.001;
errF = 1e-4;
err = 1;
frad = 220;%loading
rpm = 20000;
m = 8;
alpha = 0.1;
R0 = 0.014;
R1 = 0.028;
rpmtomms = rpm*2*pi*(R0+R1)/(2*60);
B = pi*(R0+R1)/m;
L = R1-R0;
ita = 0.01; %constant viscosity

h0 = 0.0005;

    while err > errF

        k = B*tan(alpha*pi/180)/h0;
        fu  = -(6*ita*rpmtomms*L*B^2)*(-log(k+1)+(2*k)/(k+2))
           /((k^2)*(h0^2));
        f   = fu-frad;
        h0 = h0+h0eps;
        kk = B*tan(alpha*pi/180)/h0;
        fu  = -(6*ita*rpmtomms*L*B^2)*(-log(kk+1)+(2*kk)/(kk
           +2))/((kk^2)*(h0^2));
        f1  = fu-frad;
        h0 = h0-h0eps;

        jacobian = (f1-f)/h0eps;
        h0  = h0 - 0.0009*f/jacobian;

        err = abs(f/frad);

    end
```

```
end
```

Listing 2: This is a Matlab file for finding h_0 with *fzero* command in Matlab.

```
function [h0] = Gethzerobyfzero(R0,R1,m,alpha,frad,rpm)

rpmtoms = rpm*pi*(R0+R1)/60;
B= pi*(R0+R1)/m;
L= R1-R0;
ita=0.01; %constant viscosity

f = @(h0) frad+((6*ita*rpmtoms*L*B^2)*(-log((B*tan(alpha*pi
   /180)/h0)+1)...
              +(2*(B*tan(alpha*pi/180)/h0))/((B*tan(alpha*pi
                 /180)/h0)+2))...
              /(((B*tan(alpha*pi/180)/h0)^2)*(h0^2)));

[h0] = fzero(f, [sqrt(realmin), 1E150]);

end
```

Appendix B

Listing 3: This is a Matlab file for finding friction force with brute force way in thirteen states.

```
function optimalparameter = optimalpoint(R0min,R0max,R1min,
   R1max,mmin,mmax,alphamin,alphamax)

ita = 0.01;     %constant viscosity
FRAD =
   [220;256;343;462;567;657;772;899;1012;1059;994;537;100];
RPM =
   [20000;22500;25000;27500;30000;32500;35000;37500;40000;42500;45000;47

numstates = length(FRAD);
n = 10;
optimalparameter = zeros(numstates,7);

for stateidx = 1 : numstates

    fmin = 10000000;

    for v = 1 : (n+1)
        for w = 1 : (n+1)
            for x = 1 : (n+1)
                for y = 1 : (n+1)

                    frad = FRAD(stateidx);
                    rpm = RPM(stateidx);
                    m = round(mmin + (mmax-mmin)*((v-1)/(n)))
                       ;
                    alpha = alphamin + (alphamax-alphamin)*((
                       w-1)/(n)) ;
                    R0 = R0min + (R0max-R0min)*((x-1)/(n));
                    R1 = R1min + (R1max-R1min)*((y-1)/(n));
                    h0 = Gethzerobyfzero(R0,R1,m,alpha,frad,
                       rpm);
                    B = 2*pi*(R0+R1)/(2*m);
                    k = B*tan(alpha*pi/180)/h0;
```

```
                    W = 6 * (R1 - R0) *  (-rpm * 2 * pi * (R0
                        + R1) / (2 * 60)) * ita * (-log(k+1)
                       + ((2*k)/(k+2))) / (tan(alpha * (pi /
                       180)))^2;

                    f = (R1-R0)*(-rpm* 2* pi*(R0+R1)/(2*60))*
                       ita*B*((6/(k+2))- (4*log(k+1)/k))/h0;
                        if (f < fmin)
                        fmin = f;

                        a = [W, fmin, R0, R1, m, alpha, h0];

                        end
                end
            end
        end
    end


    optimalparameter(stateidx,:) = a;
    format shortEng
    stateidx %to show which state they are running

end


end
```

Listing 4: This is Matlab file for finding friction force with GA in each state[4].

```
FitnessFunction = @objmin;
    nvars = 4;
    mmin = 3;
    mmax = 12;
    almin = 0.1;
    almax = 1;
    R0min = 0.013;
    R0max = 0.015;
    R1min = 0.020;
    R1max = 0.050;
    LB = [mmin almin R0min R1min];
    UB = [mmax almax R0max R1max];

    options = optimoptions('ga','FitnessScalingFcn',{@
       fitscalingrank }, ...
    'PopInitRange',[LB ; UB],...
    'StallGenL',10,...
    'Generations',30, ... %stopping criteria
```

```
    'PopulationSize',25,...
    'EliteCount',1, ...
    'CrossoverFraction',0.6,...
    'UseParallel','always',...
    'MigrationInterval',10,...
    'MigrationFraction',0.1,...
    'MigrationDirection','both', ...
    'plotFcn',@gaplotbestf);

%Setup for use of function ga for integer variables
[X, fval, exitflag, output, population, scores] = ga(
   FitnessFunction,nvars,[],[],[],[],LB,UB,[],[1],options);

X
fval
```

Appendix C

Listing 5: This is a function file in Matlab to run GA in Listing 7.5. This file is a function with weight function in Table (7.1).

```
function Pz = objmintry(X)

m = X(1,1);    % Range <3-12>
alpha = X(1,2); % Range <0.1-1>
R0 = X(1,3); % Range <0.013-0.015>
R1 = X(1,4); % Range <0.020-0.050>

ita=0.01; %from prof.

frad = [220;256;343;462;567;657;772;899;1012;...
        1059;994;537;100];
rpm = [20000;22500;25000;27500;30000;32500;35000;...
        37500;40000;42500;45000;47500;50000];
w = [1/13;1/13;1/13;1/13;1/13;1/13;1/13;1/13;...
        1/13;1/13;1/13;1/13;1/13]; % weight function when
           frequency at all states are equal

%state 1
f1 = @(h01) (-(6*ita*rpm(1)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h01)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h01))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h01)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h01)^2)*(h01^2)
   ))-frad(1);
h01 = fzero(f1, [sqrt(realmin), 1E150]);
k1 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h01;
F1 = (R1-R0)*(rpm(1)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k1+2))-(4*log(k1+1)/k1))/h01;

%state 2
f2 = @(h02) (-(6*ita*rpm(2)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h02)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h02))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h02)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h02)^2)*(h02^2)
   ))-frad(2);
```

```
h02 = fzero(f2, [sqrt(realmin), 1E150]);
k2 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h02;
F2 = (R1-R0)*(rpm(2)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k2+2))-(4*log(k2+1)/k2))/h02;

%state 3
f3 = @(h03) (-(6*ita*rpm(3)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h03)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h03))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h03)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h03)^2)*(h03^2)
   ))-frad(3);
h03 = fzero(f3, [sqrt(realmin), 1E150]);
k3 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h03;
F3 = (R1-R0)*(rpm(3)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k3+2))-(4*log(k3+1)/k3))/h03;

%state 4
f4 = @(h04) (-(6*ita*rpm(4)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h04)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h04))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h04)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h04)^2)*(h04^2)
   ))-frad(4);
h04 = fzero(f4, [sqrt(realmin), 1E150]);
k4 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h04;
F4 = (R1-R0)*(rpm(4)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k4+2))-(4*log(k4+1)/k4))/h04;

%state 5
f5 = @(h05) (-(6*ita*rpm(5)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h05)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h05))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h05)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h05)^2)*(h05^2)
   ))-frad(5);
h05 = fzero(f5, [sqrt(realmin), 1E150]);
k5 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h05;
F5 = (R1-R0)*(rpm(5)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k5+2))-(4*log(k5+1)/k5))/h05;

%state 6
f6 = @(h06) (-(6*ita*rpm(6)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h06)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h06))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h06)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h06)^2)*(h06^2)
   ))-frad(6);
```

```
h06 = fzero(f6, [sqrt(realmin), 1E150]);
k6 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h06;
F6 = (R1-R0)*(rpm(6)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k6+2))-(4*log(k6+1)/k6))/h06;

%state 7
f7 = @(h07) (-(6*ita*rpm(7)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h07)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h07))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h07)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h07)^2)*(h07^2)
   ))-frad(7);
h07 = fzero(f7, [sqrt(realmin), 1E150]);
k7 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h07;
F7 = (R1-R0)*(rpm(7)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k7+2))-(4*log(k7+1)/k7))/h07;

%state 8
f8 = @(h08) (-(6*ita*rpm(8)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h08)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h08))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h08)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h08)^2)*(h08^2)
   ))-frad(8);
h08 = fzero(f8, [sqrt(realmin), 1E150]);
k8 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h08;
F8 = (R1-R0)*(rpm(8)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k8+2))-(4*log(k8+1)/k8))/h08;

%state 9
f9 = @(h09) (-(6*ita*rpm(9)*2*pi*(R0+R1)/(2*60)*(R1-R0)*(2*pi
   *(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h09)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)
   /h09))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h09)+2))
   /((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h09)^2)*(h09^2)
   ))-frad(9);
h09 = fzero(f9, [sqrt(realmin), 1E150]);
k9 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h09;
F9 = (R1-R0)*(rpm(9)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k9+2))-(4*log(k9+1)/k9))/h09;clc;

%state 10
f10 = @(h010) (-(6*ita*rpm(10)*2*pi*(R0+R1)/(2*60)*(R1-R0)
   *(2*pi*(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(
   alpha*pi/180)/h010)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h010))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/
   h010)+2))/((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h010)
   ^2)*(h010^2)))-frad(10);
```

```
h010 = fzero(f10, [sqrt(realmin), 1E150]);
k10 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h010;
F10 = (R1-R0)*(rpm(10)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k10+2))-(4*log(k10+1)/k10))/h010;

%state 11
f11 = @(h011) (-(6*ita*rpm(11)*2*pi*(R0+R1)/(2*60)*(R1-R0)
   *(2*pi*(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(
   alpha*pi/180)/h011)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h011))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/
   h011)+2))/((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h011)
   ^2)*(h011^2)))-frad(11);
h011 = fzero(f11, [sqrt(realmin), 1E150]);
k11 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h011;
F11 = (R1-R0)*(rpm(11)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k11+2))-(4*log(k11+1)/k11))/h011;

%state 12
f12 = @(h012) (-(6*ita*rpm(12)*2*pi*(R0+R1)/(2*60)*(R1-R0)
   *(2*pi*(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(
   alpha*pi/180)/h012)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h012))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/
   h012)+2))/((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h012)
   ^2)*(h012^2)))-frad(12);
h012 = fzero(f12, [sqrt(realmin), 1E150]);
k12 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h012;
F12 = (R1-R0)*(rpm(12)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k12+2))-(4*log(k12+1)/k12))/h012;

%state 13
f13 = @(h013) (-(6*ita*rpm(13)*2*pi*(R0+R1)/(2*60)*(R1-R0)
   *(2*pi*(R0+R1)/(2*m))^2)*(-log(((2*pi*(R0+R1)/(2*m))*tan(
   alpha*pi/180)/h013)+1)+(2*((2*pi*(R0+R1)/(2*m))*tan(alpha*
   pi/180)/h013))/(((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/
   h013)+2))/((((2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h013)
   ^2)*(h013^2)))-frad(13);
h013 = fzero(f13, [sqrt(realmin), 1E150]);
k13 = (2*pi*(R0+R1)/(2*m))*tan(alpha*pi/180)/h013;
F13 = (R1-R0)*(rpm(13)*2*pi*(R0+R1)/(2*60))*ita*(2*pi*(R0+R1)
   /(2*m))*((6/(k13+2))-(4*log(k13+1)/k13))/h013;


Pz = -((F1 * R1 * rpm(1)/60 * w(1)) + (F2 * R1 * rpm(2)/60 *
   w(2)) + (F3 * R1 * rpm(3)/60 * w(3)) + (F4 * R1 * rpm(4)
   /60 * w(4)) + ...
     (F5 * R1 * rpm(5)/60 * w(5)) + (F6 * R1 * rpm(6)/60 * w
        (6)) + (F7 * R1 * rpm(7)/60 * w(7)) + (F8 * R1 * rpm
        (8)/60 * w(8)) + ...
```

```
(F9 * R1 * rpm(9)/60 * w(9)) + (F10 * R1 * rpm(10)/60 * w
   (10)) + (F11 * R1 * rpm(11)/60 * w(11)) + (F12 * R1 *
   rpm(12)/60 * w(12)) +...
(F13 * R1 * rpm(13)/60 * w(13)));
```